Master of Science Thesis

Design and implementation of a software defined HiperLAN/2 physical layer model for simulation purposes

L.F.W. van Hoesel

University of Twente
Department of Electrical Engineering
Chair for Signals and Systems
Enschede, The Netherlands

Supervisors: Prof. Dr. Ir. C.H. Slump
Ir. V.J. Arkesteijn
Ir. F.W. Hoeksema
Ir. R. Schiphorst

Period of work: December 2001 – August 2002

Date: August 20, 2002

Report code: SAS 032 N 02

Abstract

In this Master of Science thesis a simulation model of the HiperLAN/2 physical layer is designed and implemented. The model should provide insight in the demodulation functions that are necessary in HiperLAN/2 and it should be useful for determining channel selection and computational requirements for the software defined radio project[1] at the University of Twente. The model is implemented in Matlab Simulink and uses C++ as descriptive language.

Before the transmitted signal reaches the demodulation part in the receiver, it is distorted by the radio channel, noise, interference and the receiver hardware. These effects are modelled and their influence on the system's performance is determined. A receiver should contain –besides inverse OFDM and subcarrier demodulation– a channel estimator, a frequency offset and phase offset corrector and a symbol window tracker.

These functions have been implemented in the HiperLAN/2 receiver model and experiments have been done. The performance of the simulation model on an additive white Gaussian noise channel did match the theoretical expected performance within 0.1 dB.

Simulations also showed that good demodulation can take place with 32-bit fixed point numbers. The frequency offset and phase offset corrector showed that they can correct distortions within 1 dB of the expected performance.

[1] See [1]

Abbreviations

ACH	Access feedback CHannel
ADC	Analog to Digital Converter
ALU	Arithmetic Logic Unit
AP	Access Point
BCH	Broadcast CHannel
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
CL	Convergence Layer
DC	Direct Current
DFT	Discrete Fourier Transform
DLC	Data Link Control
EC	Error Control
ETSI	European Telecommunications Standards Institute
FCH	Frame CHannel
FEC	Forward Error Correction
FFT	Fast Fourier Transformation
HIPERLAN/2	HIgh PERformance Radio Local Area Network
I	In-phase
IDFT	Inverse Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transformation
LAN	Local Area Network
LNA	Low Noise Amplifier
MAC	Medium Access Control
MT	Mobile Terminal
OFDM	Orthogonal Frequency Division Multiplexing
PER	Packet Error Rate
PHY	PHYsical layer
Q	Quadrature
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
QPSK	Quaternary Phase Shift Keying
RCH	Random access CHannel
RLC	Radio Link Control
SDR	Software Defined Radio
VCO	Voltage Controlled Oscillator
WLAN	Wireless LAN

Symbols

Δ_f	Subcarrier spacing
ψ	Phase difference between transmitter and receiver mixers
$b_0, b_1, ...$	Bit stream in transmitter *after* FEC coding
$C_{l,n}$	Complex symbol value of OFDM symbol n carried by subcarrier l
$d_0, d_1, ...$	Bit stream in transmitter *before* FEC coding
d_f	Free distance of the the FEC coding
E_b	Bit energy $[J/bit]$
f_Δ	Frequency difference between transmitter and receiver mixers
f_{ADC}	Sample frequency of the ADC
f_{sample}	Sample rate
$g_0, g_1, ...$	Input bits of the mapping function in the transmitter
$\jmath$	Square root of minus one ($\sqrt{-1}$)
K_{mod}	Modulation type dependant normalization factor
N_0	Noise power $[W/Hz]$
N_{BPSC}	The number of bits carried on one OFDM sub carrier
N_{error}	Number of bit errors
N_{SD}	Number of data carriers
N_{SP}	Number of pilot carriers
N_{ST}	Total number of carriers
p_m	Pilot value m
R	Bite rate $[bit/s]$
R_c	Coding rate of the FEC coding
$s(t)$	Transmitted bandpass signal
$\widetilde{s_n}(t)$	Baseband version of the transmitted signal
T_{CP}	Cyclic prefix duration
T_S	OFDM symbol interval
T_U	Useful symbol part duration
W	Occupied bandwidth
$\Im\{x\}$	Imaginary part of x
$\Re\{x\}$	Real part of x

Contents

1 Introduction **6**
- 1.1 Objectives . . . 7
- 1.2 Communication system . . . 7
- 1.3 HiperLAN/2 system . . . 8
  - 1.3.1 HiperLAN/2 network . . . 8
  - 1.3.2 HiperLAN/2 protocol layers . . . 9
- 1.4 Outline . . . 11

2 Physical Layer of HiperLAN/2 Transmitter **12**
- 2.1 Introduction . . . 12
- 2.2 Performance measurements in the system . . . 14
  - 2.2.1 Bit error rate . . . 14
  - 2.2.2 Packet error ratio . . . 14
  - 2.2.3 Minimum sensitivity . . . 15
- 2.3 Scrambling . . . 15
- 2.4 Forward error correction coding . . . 16
  - 2.4.1 Bit-rate independent FEC coding . . . 17
  - 2.4.2 Bit-rate dependent FEC coding . . . 19
- 2.5 Data interleaving . . . 21
- 2.6 Mapping . . . 21
- 2.7 Orthogonal frequency division multiplexing . . . 23
  - 2.7.1 Pilot carriers . . . 23
  - 2.7.2 Modulation . . . 24
  - 2.7.3 Cyclic prefix . . . 25
- 2.8 Physical burst generation . . . 26
- 2.9 Transmission of the burst . . . 29
  - 2.9.1 Spectrum of baseband signal . . . 29
  - 2.9.2 Carrier frequency allocation . . . 30
  - 2.9.3 Bandpass signal . . . 30
- 2.10 Expected performance for AWGN channel . . . 30
- 2.11 Transmitter model implementation . . . 32
- 2.12 Conclusion . . . 34

3 Signal Distortions in the HiperLAN/2 System **35**
3.1 Introduction . . . 35
3.2 Indoor radio channel . . . 36
3.2.1 Multipath propagation mechanisms . . . 37
3.2.2 Rayleigh Channel Model . . . 38
3.2.3 Delay spread and coherence bandwidth . . . 39
3.2.4 Doppler shift and coherence time . . . 40
3.2.5 Transfer function . . . 41
3.2.6 Baseband description of the indoor radio channel . . . 41
3.2.7 Noise and interference . . . 43
3.3 Analog hardware architecture of the SDR receiver . . . 43
3.3.1 Frequency offset . . . 45
3.3.2 Phase offset and phase noise . . . 47
3.4 Sampling the signal . . . 49
3.4.1 Symbol window drift . . . 49
3.5 Digital channel selection . . . 52
3.6 Digital hardware architecture of the SDR receiver . . . 54
3.6.1 General binary number representation . . . 54
3.6.2 Integer numbers . . . 55
3.6.3 Fixed point numbers . . . 56
3.6.4 Floating point numbers . . . 56
3.6.5 Arithmetic logic unit model . . . 57
3.7 Conclusion . . . 58

4 Receiver Model Algorithms and Implementation **60**
4.1 Introduction . . . 60
4.2 Receiver architecture . . . 60
4.3 Serial to parallel conversion . . . 62
4.4 Synchronization . . . 63
4.4.1 Detecting a transmission . . . 64
4.4.2 Detecting preamble sections . . . 64
4.4.3 Tracking symbol window drift . . . 66
4.5 Prefix removal . . . 69
4.6 Frequency offset estimation . . . 69
4.6.1 Measuring frequency offset . . . 70
4.6.2 Correcting frequency offset . . . 71
4.7 Inverse orthogonal frequency division multiplexing . . . 72
4.8 Common phase offset correction . . . 72
4.9 Phase noise correction . . . 73
4.10 Channel equalization . . . 73
4.11 Demapping . . . 74
4.12 Conclusion . . . 75

5 Model Simulation Results **76**
5.1 Introduction . . . 76
5.2 Ideal channel simulation results . . . 76
5.2.1 Visualization of outputs . . . 77
5.2.2 Computational requirements of transmitter and receiver algorithms . . . 78

5.3 Comparison theoretical and simulated performance on AWGN channel . . . 81
5.3.1 Experiment configuration . . . 81
5.3.2 AWGN channel simulation results using 64-bit floating point numbers . . . 82
5.3.3 AWGN channel simulation results using 32-bit fixed point numbers . . . 83
5.3.4 AWGN channel simulation results using 16-bit fixed point numbers . . . 84
5.4 Phase offset simulation . . . 84
5.4.1 Experiment configuration . . . 84
5.4.2 Results . . . 85
5.5 Frequency offset simulation results . . . 85
5.5.1 Experiment configuration . . . 85
5.5.2 Results . . . 87
5.6 Signal generator – scope channel results . . . 87
5.6.1 Results . . . 89
5.7 Conclusion . . . 89

6 Conclusions and Recommendations **91**
6.1 Conclusions . . . 91
6.2 Recommendations . . . 94

Bibliography **95**

Chapter 1

Introduction

A *software defined radio* is a radio architecture, that performs as many as possible demodulation functions using *software*, assisted by an as general as possible, but *minimal* hardware structure. Conventional radio architectures use a large amount of hardware, that is *specially* designed for the specific radio application.

A software defined radio has two advantages over a conventional radio. Users will become *independent* of region bounded communication system specifications, by simply downloading software code for a specific communication system and –a probably more realistic advantage– manufactures will only have to design *one* receiver architecture, that can be used for many radio applications. This saves a large amount of design costs.

The software defined radio (*SDR*) project[1] at the University of Twente is currently investigating the feasibility of a software defined radio system. This is done by developing a *demonstrator*, that is capable of demodulation two communication standards: *HiperLAN/2*, a high-speed wireless local area network (*WLAN*) standard, and *Bluetooth*, a low-cost and low-speed personal area network (*PAN*) standard.

The necessary functions for demodulation and the requirements for a good reception are currently assessed for the two communication standards. These results will be used to design an *analog front-end*, *digital channel selection filters* and a *demodulator*.

In the following section, the objectives of this report are outlined. HiperLAN/2 is the standard, that is under investigation in this report.

In section 1.2 an introduction to communication systems will be given. Section 1.3 discusses the HiperLAN/2 communication system shortly. Section 1.4 sketches the outline of this report.

[1] See [2] and [1].

1.1 Objectives

The objective of this Master of Science thesis is to design, implement and verify a *simulation model* of the physical layer in HiperLAN/2.

The model should provide insight in the demodulation functions that are necessary in HiperLAN/2 and it should be useful for determining channel selection and computational requirements. Another important objective is that the model should be able to study effects of signal distortions by –for example– the radio channel and the effects of computational noise.

The model must also be able to generate *test signals* for the analog front-end and digital channel selection filters and the model must be able demodulate captured signals from the analog front-end. The model must also be easy expandable; it must be easy to test new or improved demodulation algorithms and to determine their performance.

The simulation model must be implemented in Matlab Simulink (see [3]). The model will be written in the programming language *C++* (see [4]).

1.2 Communication system

A communication system transfers *information* from a source to a user of the information. In general a communication system exists of five parts: a source of information, a transmitter of the information, a channel, a receiver and finally a user of the information. This system is depicted in figure 1.1.

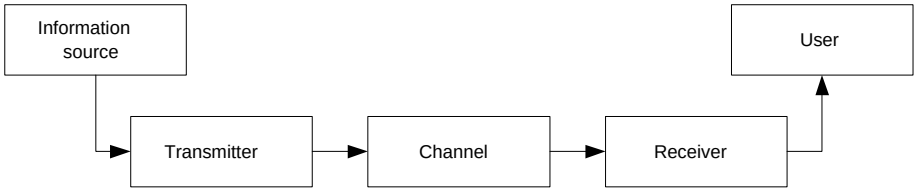


Figure 1.1: A communication system

Normally the user of the information is not interested in the manner how the information is delivered, but merely in the information itself and the accuracy of the information. Modern communication systems meet this approach. The systems contain many *layers* that communicate with each other. Each individual layer can be seen as an information source at the transmitter side and an information user at the receiver side.

Why use this layer approach? The communication systems nowadays become more and more complex. For instance, the channel must be divided along many transmitters and receivers. Another example is that there are many types of information messages, that all should be presented to the user in a different way. Those different message types are most likely transmitted in the same

way. So some functions in the system are common to all message types, while others differ. The division of a communication system in layers makes the system comprehensible to designers, it makes the system testable in sections and it makes the system easily expandable (see [5]).

1.3 HiperLAN/2 system

In this report a model is made of the high performance radio local area network (*HiperLAN/2*) *physical layer*. HiperLAN/2 (see [6]) is a standard for a *wireless local area network* (*WLAN*), that has been developed by the *European telecommunications standards institute* (*ETSI*). HiperLAN/2 provides a high transmission speeds from 6 to 54 Mbit/s. This speed is necessary to meet the actual requirement for –in example– internet access and hence it is expected that the standard will broaden its market share in the next few years.

The HiperLAN/2 standard defined by ETSI has an American counterpart: IEEE 802.11. Both standards use more or less the same physical layer, but differ in other layers (see [6]).

In the following sections the network architecture of HiperLAN/2 is outlined, followed by an introduction to the protocol layers in HiperLAN/2.

1.3.1 HiperLAN/2 network

The HiperLAN/2 wireless network has two types of communication devices:

- Mobile Terminals (MT)
- Access Points (AP)

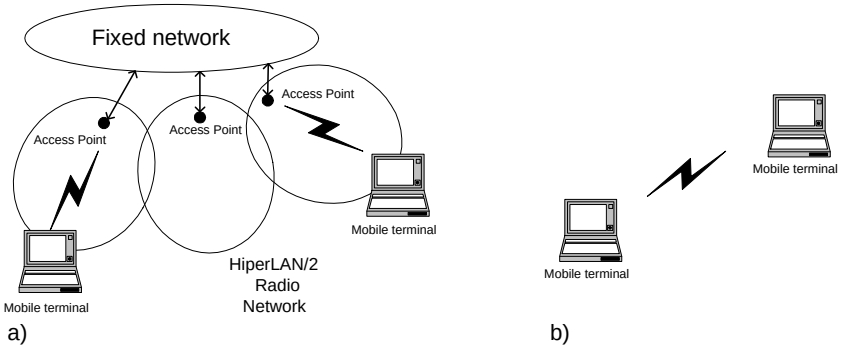


Figure 1.2: A typical HiperLAN/2 network. a) Mobile terminals communicate with a fixed network via access points. b) Mobile terminals communicate directly with each other

A typical HiperLAN/2 network is depicted in figure 1.2. The MTs communicate with a fixed network via APs. An example of such a fixed network is a local area

network (LAN). HiperLAN/2 supports packet based (ethernet, internet protocol etc), as well as cell based (asynchronous transfer mode etc) communication (see [5] and [6]). At a given time an MT will only communicate with one AP.

The MT may move around freely in the the HiperLAN/2 network. The system will ensure that an MT always gets the best possible transmission performance. This transmission performance is highly dependent on distortions that occur in the radio link (see chapter 3).

Besides connections between AP and MT, the system also supports direct connections between two MTs (see [6]). This direct link is specially useful in home situations.

1.3.2 HiperLAN/2 protocol layers

In figure 1.3 the protocol model of HiperLAN/2 is outlined (see [6]). The system has three main protocol layers: the physical (PHY) layer, the data link control (DLC) layer and the convergence layer CL.

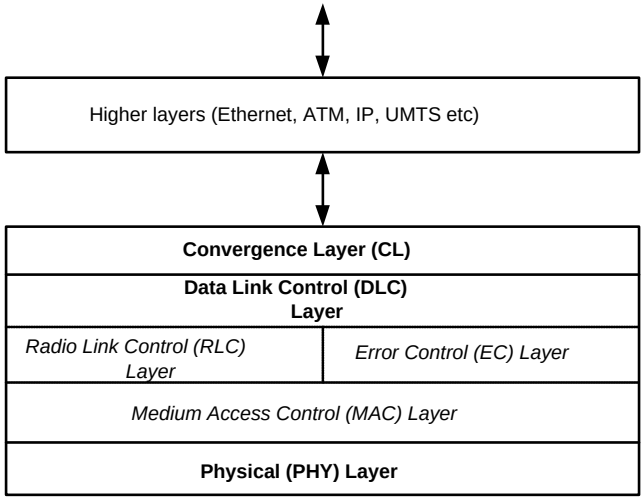


Figure 1.3: HiperLAN/2 protocol model (see [6])

The task of the *physical layer* in HiperLAN/2 is to modulate *bits* that originate from the *data link control layer* on the transmitter side and to demodulate them on the receiver side (see chapter 2). The transmission format on the physical layer is a *burst*, which consists of a *preamble* and a *data* part.

The frequency spectrum available to HiperLAN/2 is divided into 19 so called *channels*. In this report we will refer to those channels as *radio channels*. Each of those radio channels has a frequency bandwidth of 20 MHz.

Orthogonal frequency division multiplexing ($OFDM$) has been chosen as modulation technique in HiperLAN/2 , because it has a good performance on

an *indoor radio channel*. OFDM is a special kind of *multicarrier modulation*. The modulation technique divides the high data rate information in several parallel bit streams and each of those bit streams modulates a separate *subcarrier*. In this way the radio channel is divided into several independent subchannels, which enables each carrier to support data at a low rate. This enables OFDM to have a good performance on highly dispersive channels.

The modulation technique has other benefits: it deals efficiently with the spectrum, since the subcarriers are spaced at minimal distance to each other.

Objects in the proximity of the radio transmitter or receiver can cause that the receiver receives multiple, delayed and attenuated versions of the transmitted signal. HiperLAN/2 OFDM is also designed to deal with channels that have a *delay spread* up to 250 ns (see chapter 3).

The physical layer transmits 52 subcarriers in parallel per radio channel. Not all subcarriers contain information that will be delivered to the data link control layer. Four of the 52 subcarriers are used to transmit *pilot tones*. Those pilots assist the demodulation in the receiver.

The *data link control* (*DLC*) layer takes care of *logical links* between APs and MTs. The layer is divided into three sublayers: medium access control (*MAC*) layer, error control (*EC*) layer and radio link control (*RLC*) layer. Below these sublayers will be discussed shortly.

The MAC layer is used to organize the use of the radio link ("whose turn it is"). The control of the medium is centralized to the AP. It tells the MTs when they are allowed to transmit data –this is called *uplink* communication– and when the AP transmits data for a specific MT –this is called *downlink* communication. The sublayer takes also care of dividing the available medium resources to the needs of the MTs.

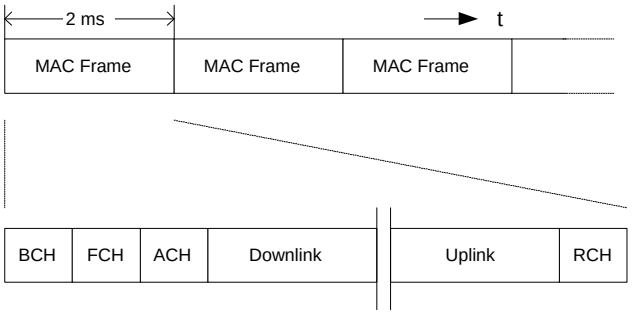


Figure 1.4: The HiperLAN/2 MAC frame has a fixed duration of 2 ms and is divided in six *logical* channels: a broadcast channel (BCH), a frame control channel (FCH), an access feedback channel (AFC), a downlink phase, a uplink phase and a random access channel (RCH. see [6])

The MAC protocol is based upon *time-division duplex* and *dynamic time-division multiple access*. This means that time has been divided in so called MAC frames, which allow for simultaneous communication from AP to MT and visa versa within that frame (see figure 1.4). The duration for uplink and

downlink phase is dynamically established in the MAC frame depending on the needs in the network.

The MAC frame is divided in six *logical channels*. Each MAC frame starts with the transmission of *control data* to the MTs. This takes place in the so called *broadcast channel* (*BCH*). It informs MTs about transmission powers and timings of other logical channels in the frame. The BCH part is followed by the *frame control channel* (*FCH*), it contains an exact description of how resources have been allocated in the current frame for uplink and downlink phase. The following channel, the *access feedback channel* (*ACH*), contains information about previous access attempts made in the *random access channel* (*RCH*). Next the downloading and uploading takes place and finally MTs are allowed to ask for transmission resources in the so called RCH. Each MAC frame has a fixed duration of 2 ms.

The *error control* (*EC*) protocol increases the reliability of the transmission over the data link. It detects bit errors in the receiver and asks for a retransmission of the data. The quality of service (*QoS*) can be adapted to the needs of the transmitted data. For example voice transmissions are sensitive to delays and hence more missing data will be allowed to keep the delay to a minimum.

The top layer in the HiperLAN/2 standard is the *convergence layer* (*CL*). This layer forms a bridge between HiperLAN/2 and higher layers –like IP– that use HiperLAN/2 as transport mechanism. Its main function is to convert packets from those higher layers to packets that can be used in the HiperLAN/2 system. It makes the HiperLAN/2 connection suitable for a diversity of fixed networks like Ethernet, IP, ATM, UMTS and many others (see [5] and [6]).

1.4 Outline

Chapter 2 of this report discusses the HiperLAN/2 physical layer in the transmitter and the theoretical expected performance of the system on an *additive white Gaussian noise* (*AWGN*) channel. The result of this chapter is a verified model of the physical layer of the HiperLAN/2 transmitter, that is capable to generate test signals.

In chapter 3 the signal distortions in the (indoor) radio channel, that guides the radio waves from transmitter to receiver, are discussed. The analog front-end and digital channel selection filters disturb the received signal too. Effects that occur, will be discussed. Finally chapter 3 describes a method to simulate computational noise.

The results from 2 and 3 are used to design and to implement a receiver model, that is capable of correcting the signal distortions (partly). The source code of the models is printed in [7].

In chapter 5 some simulation results are presented.

Chapter 6 summarizes the conclusions of this report and discusses some future research topics for HiperLAN/2 used on the SDR project's architecture.

Chapter 2

Physical Layer of HiperLAN/2 Transmitter

2.1 Introduction

The HiperLAN/2 physical layer provides transportation mechanisms of bits between the data link control (*DLC*) layer in transmitter and receiver. The ETSI documentation [8] defines the physical layer in the transmitter with seven functions:

- Scrambling of the binary input stream
- Forward error correction coding
- Interleaving
- Mapping
- Modulation using orthogonal frequency division multiplexing
- Physical burst generation
- Transmitting of the burst

In figure 2.1 the data flow between the seven functions is outlined. At point *A* in the figure, the output bits of the DLC layer enter the physical layer. Those bits are scrambled, forward error correction coding is applied and the bits are interleaved. This results in a bit group at *D* called *raw* bits. The raw bits are mapped to complex symbols by the mapping function. So at *E* and further on, the signals between the entities are *complex numbers* instead of bits.

The complex subcarrier values at *E* are modulated with a modulation technique called orthogonal frequency division multiplexing (*OFDM*) and this technique will be discussed in section 2.7. The resulting OFDM symbols at *F* are

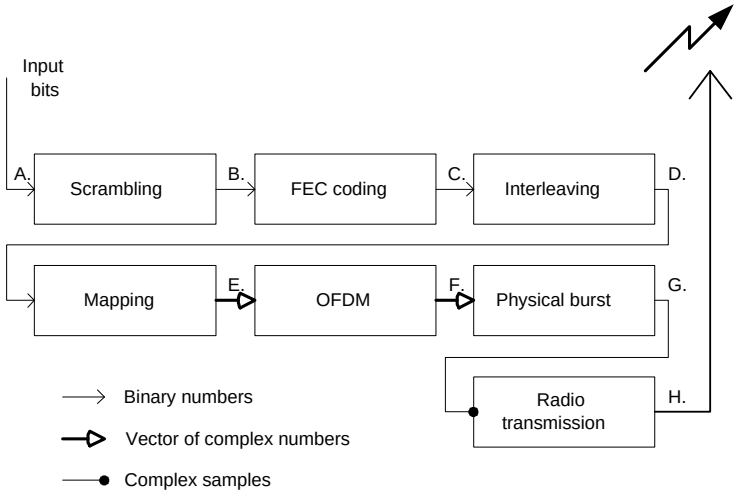


Figure 2.1: Data flow between the seven functions of the HiperLAN/2 physical layer in the transmitter (see [8])

proceeded by a *preamble*; a sequence of known OFDM symbols. This preamble is transmitted before the physical layer starts transmitting the OFDM symbols containing the actual data.

At G we find complex time samples that represent the preamble and the OFDM symbols. Finally the complex time samples are converted to analog I- and Q-signals and mixed to a specific carrier frequency in the radio transmission function.

The tasks of the physical layer on the transmitter side (see figure 2.1) will be discussed in detail in the following sections. We will see by "reverse engineering" why certain choices are made in the physical layer of the HiperLAN/2 system. This gives useful insight in the system for designing a HiperLAN/2 receiver.

In section 2.9 the spectrum of the transmitted HiperLAN/2 signal will be discussed. We are interested in the performance of the system in case a certain distortion (for example noise) is present. Chapter 3 discusses the distortions present in the transmitter, channel and receiver system, but in section 2.10 we will already have a closer look to the theoretical performance of the system. In section 2.2 the definitions of *bit error rate*, *bit energy* and *packet error ratio* will be explained. The results of section 2.10 will be used in chapter 5 to make a comparison between the implemented system and the theoretical performance for simulation model validation purposes.

The result of this chapter is a *simulation model* of the physical layer of the HiperLAN/2 transmitter, that can be used for testing receiver algorithms. The implementation of the model is discussed in section 2.11. Chapter 4 will discuss the receiver model. Below, the transmitter functions will be further discussed. Table 2.1 shows what signal names will be used throughout those

function descriptions.

Table 2.1: Signal names in the transmitter

Signal point in figure 2.1	Signal name
B	$d_0, d_1, ...$
C	$b_0, b_1, ...$
D	$g_0, g_1, ...$
E	$C_{l,n}$
G	$\widetilde{s_n}(t)$ or $\widetilde{s_n}[m]$
H	$s(t)$

2.2 Performance measurements in the system

2.2.1 Bit error rate

The performance of a transmitter, channel and receiver system is expressed with the *bit error rate* (BER), defined as:

$$BER = \frac{N_{error}}{N_{Total}} \tag{2.1}$$

where N_{error} is the number of bits that are wrongly received and N_{total} is the total number of bits received. This measurement of bit error probability is plotted against the cause that bits are received wrongly:

$$\frac{E_b}{N_0} = \frac{P_s}{P_n}\frac{W}{R} \tag{2.2}$$

a ratio between the average energy that is put in the signal per transmitted bit in $[J/bit]$ and the power of noise $[W/Hz]$. In equation 2.2, P_s is the average power of the transmitted signal in $[W]$, P_n is the average noise power, R is the number of bits per second and W is the bandwidth used by the signal[1]. The spectral density of the white noise N_0 is measured at the input stage of the detector (see [10]), after channel selection filters (see section 3.5 and figure 3.1).

2.2.2 Packet error ratio

The performance can also be expressed in *packet error ratio* (PER); a ratio between the correctly received number of packets and the totally transmitted number of packets. A packet is a group of bits and HiperLAN/2 uses a fixed packet size of 54 bytes (see [8]). Note that *one* incorrect received bit in the group of 54 bytes marks the packet to be received incorrectly.

The PER can be calculated from BER as follows (see [5]):

$$PER = 1 - (1 - BER)^{54*8} \tag{2.3}$$

[1]This is known as the *occupied bandwidth* (see [9])

2.2.3 Minimum sensitivity

The HiperLAN/2 standard defines a *minimum sensitivity*[2] for the receiver, where the packet error rate of the system needs to be 10% or less for packets of 54 bytes (see [8]). The system needs a bit error rate of $P_b = 2.4\,10^{-3}$ to reach a packet error rate of 10% (see section 2.2.2).

In this report we will determine what E_b/N_0-ratio is necessary to reach a PER of 10%.

2.3 Scrambling

This function (see figure 2.2) scrambles the incoming information bits (A in figure 2.1) that are generated by the data link control layer. Its generator polynomial is given by:

$$X_7 \oplus X_4 \oplus 1 \tag{2.4}$$

Where $X_{1..7}$ represents the state of the scrambler. Note that "$\oplus$"[3] denotes a modulo two adder.

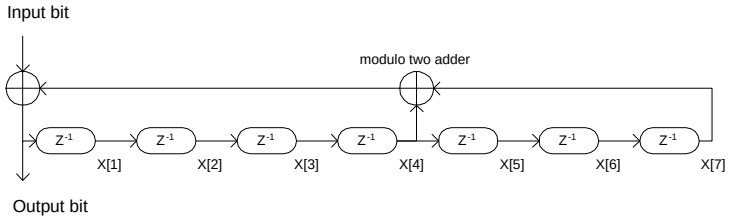


Figure 2.2: Scrambler of HiperLAN/2 physical layer [8]

The scrambler is initialized at different moments, depending on what type of data is transmitted. The different data types and their relation with the scrambler initialization moments are discussed in detail in [8] and will not be discussed in this report.

The initial state of the scrambler depends on the first four bits in the broadcast channel (BCH). Those four bits $n_4, \ldots, n_1$ represent the number of the frame that is currently transmitted. The scrambler vector $X_{1..7}$ is initialized according to table 2.2.

Table 2.2: Initialization of the scrambler vector

	$X[1]$	$X[2]$	$X[3]$	$X[4]$	$X[5]$	$X[6]$	$X[7]$
Value	n_1	n_2	n_3	n_4	**1**	**1**	**1**

[2]Power level measured at the antenna of the receiver.
[3]This function is also known as an "*exclusive OR*"

The purpose of the scrambler in the HiperLAN/2 system is to limit the number of consecutive **1**'s (or **0**'s) to about three. In this way the output symbols of the mapping entity (see section 2.6) will change although a bit stream of -in example- all **1**'s is transmitted. This randomizing of the output symbols results in a possible lower bit error rate at the receiver.

Decoding of a scrambled bit stream, can be done with an equal scrambler, if it is equally initialized.

The scrambler will not be implemented in the transmitter model, because we will use only a random bit source to obtain simulation results. Future work should implement this function.

2.4 Forward error correction coding

The next operation in the physical layer of HiperLAN/2 , is forward error correction (*FEC*) coding. It inserts "redundant" bits in the scrambled bit stream, in such way that error correction can be applied in the receiver, in a systematic manner.

Table 2.3: Bit-rate modes of HiperLAN/2 ; section 2.6 explains the subcarrier modulation

Mode	Data bit-rate [Mbit/s]	Subcarrier modulation	Code rate R_c
A	6	BPSK	1/2
B	9	BPSK	3/4
C	12	QPSK	1/2
D	18	QPSK	3/4
E	27	16QAM	9/16
F	36	16QAM	3/4
G	54	64QAM	3/4

The HiperLAN/2 system is able to operate with various bit-rates (see table 2.3 and [8]). The code rate R_c is defined as:

$$R_c \triangleq \frac{Input\ bitrate}{Output\ bitrate} \tag{2.5}$$

where *Input and Output bit-rate* are the input and output bit-rate of the *FEC coding*.

The transmitter and receiver decide per transmission burst what bit-rate is actually used. The ETSI standard [8] does not define the procedure for choosing a mode.[4] For simplicity we will assume in this report that the bit-rate is fixed during a simulation session and that no dynamical mode changes are made.

The forward error correction coding of the HiperLAN/2 system can be divided in a bit-rate mode independent part and a bit-rate dependent part. First

[4] In [11] a method is proposed, which maximizes the data bit throughput of the system, for a given signal to noise ratio.

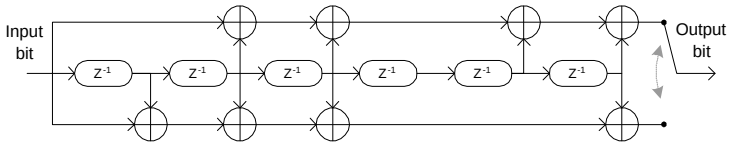


Figure 2.3: Convolutional encoder of HiperLAN/2 (see [8])

the bit-rate independent part of the FEC coding will be discussed, followed by the bit-rate mode-dependent part.

2.4.1 Bit-rate independent FEC coding

The basis of the bit-rate independent FEC coding is a nonsystematic convolutional encoder (see figure 2.3) with *constraint length* seven (see [12]). The convolutional encoder generates two bits per input bit, hence it has a code rate $R_c = 1/2$. The two output values are dependent on the current input bit and six input bits in the past. The output bits are combined into one bit stream, with twice the rate of the input bit stream.

The main difference between convolutional coding and block coding is that block coding divides the input stream in sections, while convolutional coding creates a bit stream (see [12]). Both methods of coding use only a small fraction of possible output sequences as codeword sequences. The analysis of convolutional coding closely matches the analysis of block coding (see [12]).

The convolutional encoder used in HiperLAN/2 can correct up to nine bit errors in combination with an ideal decoder, from the moment a bit error occurs until the state of the decoder matches is correct again. An important measure for a specific convolutional encoder is called the *free distance* d_f. It tells at how many incorrectly received bits, the first bit errors start to emerge at the output of the (ideal) decoder. However, for more than nine bit errors in the received bit stream, the bit error probability can greatly be reduced, compared to no error correction. Of course this performance reduces the data bit throughput from transmitter to receiver and costs processing power in the receiver for decoding.

In this report we will not implement the FEC coding, but we will deduce a lower limit (the real performance can be better) for the performance of the convolutional code based on the analysis presented in [12]. For this analysis we assume that the transmitted bit sequence (the output of the convolutional encoder) is **0000**.... . This sequence is called the *zero path*. The following analysis is valid for all other transmitted sequences. The reason why this zero path sequence is chosen, is that bit errors can easily be counted with an *accounting function* (see [12]), because every received **1** means a bit error.

Assume that the probability that a transmitted **0** is received as a **1**, is p_e. A *decoding error* is made when the received bit sequence is closer to another valid output combination of the convolutional encoder, than to the correct zero path.

The probability that a decoding error occurs at Hamming distance[5] k, is

$$P_k = \sum_{i=\frac{k+1}{2}}^{k} \binom{k}{i} {p_e}^i (1-p_e)^{k-i} \tag{2.6}$$

for k is odd and

$$P_k = \sum_{i=\frac{k}{2}+1}^{k} \binom{k}{i} {p_e}^i (1-p_e)^{k-i} + \frac{1}{2}\binom{k}{\frac{k}{2}} {p_e}^{\frac{k}{2}} (1-p_e)^{\frac{k}{2}} \tag{2.7}$$

for k is even.

The most right term in equation 2.7 represents the fact, that the received sequence is at equal Hamming distance to the correct path and to the wrong path. We assume that in half of the cases the correct path is chosen, hence the term is proceeded by "$\frac{1}{2}$".

The total probability of a decoding error is given by:

$$P_E < \sum_{k=d_f}^{\infty} a_k P_k \tag{2.8}$$

With a_k is the number of paths associated with Hamming distance k from the correct path and d_f is the free distance, for which all errors can be corrected.

In practice we are more interested in the probability of a *bit error*, instead of the probability of a *decoding error*. The bit error probability is given by:

$$P_b < \sum_{k=d_f}^{\infty} c_k P_k \tag{2.9}$$

With c_k is the number of bit errors associated with Hamming distance k from the correct path.

The accounting function can be used to determine the coefficient c_k (see [12]). If we assume that the decoder makes its decision before $d_f + 8$ received bits, the following c_k's are found (see table 2.4).

Table 2.4: Associated bit errors c_k with an incorrect path at Hamming distance k for the convolutional encoder used in HiperLAN/2 (see [12] page 508)

k	$d_f = 10$	$d_f + 1$	$d_f + 2$	$d_f + 3$	$d_f + 4$	$d_f + 5$	$d_f + 6$	$d_f + 7$
c_k	36	0	211	0	1404	0	11633	0

The *Viterbi* algorithm can be used for optimal decoding of a convolutional encoded bit stream. A good description of this algorithm is given in [12].

[5]Hamming distance is the number of different bits between the transmitted sequence and the received sequence.

2.4.2 Bit-rate dependent FEC coding

The section above describes the bit-rate independent FEC coding. The bit-rate dependent FEC coding punctures the output bit stream of the convolutional encoder in such way, that the code rate changes to $R_c = 3/4$, $9/16$ or remains $1/2$ (see [8] and table 2.3).

Consider the output bit stream of the convolutional encoder: $b_0 b_1 ... b_n$, with b_0 the first transmitted bit. In table 2.5 the applied puncturing is explained. Note that some bits are *not* transmitted. Before using a convolutional decoder –like the Viterbi algorithm– in the receiver, some extra bits must be inserted in the bit stream to make the codeword equal to its original size ($R_c = 1/2$).

Table 2.5: Bit-rate mode dependent puncturing. The **bold** values are placed in the output bit stream. The input bit stream of the FEC coding is denoted as $d_0, ...$ and the output of the convolutional encoder as $b_0, ...$

$R_c = \frac{1}{2}$

Input bit	d_0
Output bits	$\mathbf{b_0}$
	$\mathbf{b_1}$

$R_c = \frac{9}{16}$

Input bit	d_0	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8
Output bits	$\mathbf{b_0}$	$\mathbf{b_2}$	$\mathbf{b_4}$	$\mathbf{b_6}$	$\mathbf{b_8}$	$\mathbf{b_{10}}$	$\mathbf{b_{12}}$	$\mathbf{b_{14}}$	b_{16}
	$\mathbf{b_1}$	$\mathbf{b_3}$	$\mathbf{b_5}$	$\mathbf{b_7}$	b_9	$\mathbf{b_{11}}$	$\mathbf{b_{13}}$	$\mathbf{b_{15}}$	$\mathbf{b_{17}}$

$R_c = \frac{3}{4}$

Input bits	d_0	d_1	d_2
Output bits	$\mathbf{b_0}$	$\mathbf{b_2}$	b_4
	$\mathbf{b_1}$	b_3	$\mathbf{b_5}$

The puncturing of the convolutional coding makes the performance of the FEC coding worse. In the receiver a guess should be made what bit value should be inserted for the bits not transmitted. Usually is assumed, that the probability that a missing bit is a **1**, equals the probability that it is a **0**. Hence either one of them can be chosen.

Tables 2.6 and 2.7 give the associated bit errors c_k with a incorrect path at Hamming distance k for the convolutional encoder for coding rates of $R_c = 3/4$ and $9/16$ and figure 2.4 shows the performance of FEC coding with $R_c = 1/2$, $9/16$ and $3/4$.

From figure 2.4 can be concluded that the usage of the HiperLAN/2 convolutional encoder is only useful, when the raw bit error probability is smaller than $p_e < 0.07$ for $R_c = 1/2$, $p_e < 0.04$ for $R_c = 9/16$ and $p_e < 0.03$ for $R_c = 3/4$.

The FEC coding will not be implemented in the transmitter nor will an error correction decoder be implemented in the receiver model discussed in chapter 4. Future work should implement these functions. We will use the results of this

Table 2.6: Associated bit errors c_k with a incorrect path at Hamming distance k for the FEC coding rate $R_c = \frac{3}{4}$ (taken from [11], table 3)

k	$d_f = 5$	$d_f + 1$	$d_f + 2$	$d_f + 3$	$d_f + 4$	$d_f + 5$
c_k	2.67	10.33	53.33	297.33	1504	7769

Table 2.7: Associated bit errors c_k with a incorrect path at Hamming distance k for the FEC coding rate $R_c = \frac{9}{16}$ (taken from [11], table 4)

k	$d_f = 7$	$d_f + 1$	$d_f + 2$	$d_f + 3$	$d_f + 4$	$d_f + 5$
c_k	2	10.9	35.7	117.8	342.3	1172.2

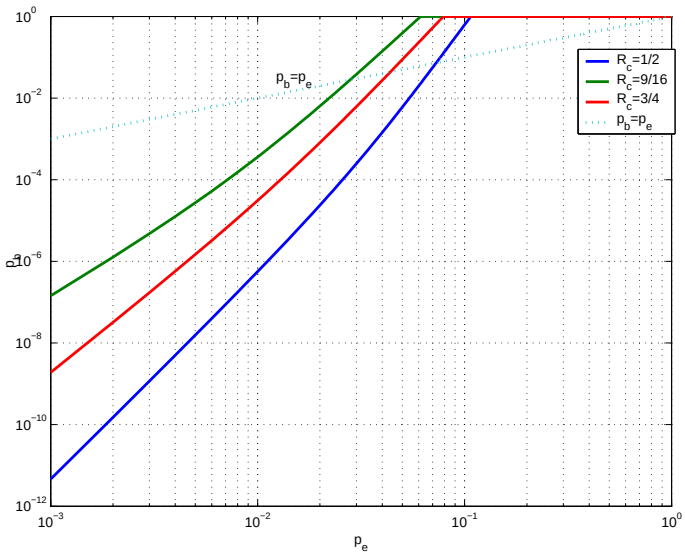


Figure 2.4: Performance of forward error correction coding ($R_c = \frac{1}{2}, \frac{9}{16}$ and $\frac{3}{4}$)

section to determine the raw bit error rate necessary to fulfill the minimum sensitivity requirement defined in [8].

2.5 Data interleaving

This entity ensures that adjacent bits of the encoded bit stream will not be modulated on adjacent OFDM subcarriers and that adjacent coded bits are mapped to alternately higher or lower significant positions in the mapping constellations. This functionality is performed by two *permutations*:

- Let k bet the index of bits at the input, let i be the bit index after this permutation and let N_{BPSC} be the number of bits mapped per OFDM subcarrier. The first permutation is given by (see [8]):

$$i = \frac{N_{BPSC}}{16}(k \mod 16) + floor\left(\frac{k}{16}\right) \tag{2.10}$$

  with $k = 0, 1, \ldots, N_{BPSC} - 1$.

- Let i be the bit index after the first permutation and let j be the index after this permutation. The second permutation is given by:

$$j = s\ floor\left(\frac{s}{i}\right) + \left(i + N_{BPSC} - floor\left(\frac{16i}{N_{BPSC}}\right) \mod s\right) \tag{2.11}$$

  with $i = 0, 1, \ldots, N_{BPSC} - 1$ and $s = \max(\frac{N_{BPSC}}{2}, 1)$.

The data interleaver will not be implemented in the transmitter model, because we will use only a random bit source to obtain simulation results. Future work should implement this function.

2.6 Mapping

This entity maps groups of N_{BPSC} bits to complex subcarrier values and applies a normalization factor (to achieve same average power for all modulation types). The input bits are modulated with binary phase shift keying ($BPSK$), quaternary phase shift keying ($QPSK$), 16 quadrature amplitude modulation ($16QAM$) or 64 quadrature amplitude modulation ($64QAM$). In tables 2.8, 2.9, 2.10 and 2.11 the used Gray coded constellations (see [8]) are shown (g_0 represents the first arriving bit in this entity). The modulation type does not change within one transmitted burst.

After applying the mapping the to the input bits, the complex result is multiplied with a modulation type dependent constant (see table 2.12). This results in a complex subcarrier value:

$$C = K_{mod}\,(\,\alpha + \jmath\,\beta\,) \tag{2.12}$$

Table 2.8: Binary phase shift keying ($N_{BSPC} = 1$)

g_0	Real output α	Complex output β
0	-1	0
1	1	0

Table 2.9: Quaternary phase shift keying ($N_{BSPC} = 2$)

g_0	Real output α	g_1	Complex output β
0	-1	**0**	-1
1	1	**1**	1

Table 2.10: Quadrature amplitude modulation (16) ($N_{BSPC} = 4$)

$g_0 g_1$	Real output α	$g_2 g_3$	Complex output β
00	-3	**00**	-3
01	-1	**01**	-1
11	1	**11**	1
10	3	**10**	3

Table 2.11: Quadrature amplitude modulation (64) ($N_{BSPC} = 6$)

$g_0 g_1 g_2$	Real output α	$g_3 g_4 g_5$	Complex output β
000	-7	**000**	-7
001	-5	**001**	-5
011	-3	**011**	-3
010	-1	**010**	-1
110	1	**110**	1
111	3	**111**	3
101	5	**101**	5
100	7	**100**	7

Table 2.12: Modulation type dependent value of the normalization factor

Modulation type	K_{mod}
BPSK	1
QPSK	$\frac{1}{\sqrt{2}} \approx 0.70711$
16QAM	$\frac{1}{\sqrt{10}} \approx 0.31623$
64QAM	$\frac{1}{\sqrt{42}} \approx 0.15430$

In groups of $N_{SD} = 48$ subcarrier values will be used by the following entity –orthogonal frequency division multiplexing– to create complex baseband samples.

In section 2.10 a comparison will be made between the expected bit error rates between "regular" BPSK, QPSK, or QAM systems and an OFDM system using the same subcarrier mapping techniques.

The mapping function is implemented in the transmitter model.

2.7 Orthogonal frequency division multiplexing

This function has the following tasks:

- Inserting pilot subcarriers
- Applying orthogonal frequency division multiplexing
- Inserting cyclic prefix

These three functions are implemented in the transmitter model presented in section 2.11. In the next sections the tasks will be discussed.

2.7.1 Pilot carriers

Four out of the 52 HiperLAN/2 subcarriers are used to transmit a known sequence. Those four subcarriers are called *pilot carriers* (see table 2.13).

The value p_m is element m from the sequence (see [8]):

$$p_{0\ldots126} = \{1, 1, 1, 1, -1, -1, -1, 1, -1, \ldots, -1, -1\} \tag{2.13}$$

With:

$$m = n \mod 127 \tag{2.14}$$

Where n is the number of the current OFDM symbol in the transmission burst.

The pilot sequence in equation 2.13 can be created with the same generation polynomial as in equation 2.4. It should be initialized with all **1**'s and a **1** in the polynomial outcome should be replaced by "-1" and a **0** should be replaced with "1" (see [8]).

Although the transmission of pilot carriers consumes energy, the usage of pilots is very useful, because it gives the receiver information about the channel; it enables equalization.

Table 2.13: Subcarriers used to transmit known pilot values

Subcarrier number l	Value
-21, -7, 7	p_m
21	$-p_m$

Table 2.14: HiperLAN/2 OFDM parameters

Parameter		Value	
Sampling rate	f_{sample}	20 MHz	$(\triangleq 1/T)$
Symbol interval	T_S	$4.0\mu s$	$(= 80 \cdot T)$
Useful symbol part duration	T_U	$3.2\mu s$	$(= 64 \cdot T)$
Cyclic prefix duration	T_{CP}	$0.8\mu s$	$(= 16 \cdot T)$
Number of data carriers	N_{SD}	48	
Number of pilot carriers	N_{SP}	4	
Total number of carriers	N_{ST}	52	
Subcarrier spacing	Δ_f	0.3125 MHz	$(= 1/T_U)$

2.7.2 Modulation

After mapping the input bits of the physical layer to $N_{SD} = 48$ complex subcarrier values and after the insertion of $N_{SP} = 4$ pilot carriers, the resulting $N_{ST} = 52$ complex subcarrier symbols are converted to 64 complex time samples, that represent the useful data part of an OFDM symbol. This operation is done by the following equation:[6]

$$\widetilde{s_n}(t) = \begin{cases} \sum_{l=-\frac{N_{ST}}{2}}^{\frac{N_{ST}}{2}} C_{l,n} e^{j2\pi l \Delta_f (t - T_{CP} - nT_S)} & , nT_S \leq t < (n+1)T_S \\ 0 & , \text{else} \end{cases} \tag{2.15}$$

The meaning of the symbols and their values are explained in table 2.14. The system does not output a direct current (DC) component ($C_{0,n} = 0$). Note that the subcarriers are *orthogonal* to each other in the interval T_U.

The OFDM operation is known as "applying a uniform synthesis inverse discrete Fourier transform ($IDFT$) filter bank" (see [13]). An often used implementation for the equation above is the inverse fast Fourier transformation ($IFFT$) algorithm. The result of this algorithm is a sampled version of the time continue equation above. We will have a closer look at this practical implementation of this equation.

The ETSI documentation [8] proposes a sample frequency $f_{sample} = 20$ MHz. In that case one OFDM symbol -the cyclic prefix included- has a duration T_S of 80 samples. The useful data part T_U has a duration of 64 samples. Assume that we calculate OFDM symbol $n = 0$ and $0 \leq t < T_U$, then the sampled version of

[6] Note that the subcarrier values actually are the *spectral contents* of the time signal.

equation 2.15 results in:

$$\widetilde{s_n}[m] \triangleq \widetilde{s_n}(t)\Big|_{t=\frac{m}{f_{sample}}} = \sum_{l=-\frac{N_{ST}}{2}}^{\frac{N_{ST}}{2}} C_{l,n} e^{j2\pi l \Delta_f \frac{m}{f_{sample}}} \tag{2.16}$$

with $m = 0, 1, 2, .., T_U f_{sample}$-1. equation 2.16 can be written as an inverse discrete Fourier transformation of the subcarrier values. The IDFT is defined as:

$$F[y] = \frac{1}{N} \sum_{x=0}^{N-1} f[x] e^{j2\pi \frac{xy}{N}} \tag{2.17}$$

with x and y=0,1,2,...,N-1. Thus

$$\widetilde{s_n}[m] = \sum_{x=0}^{N-1} f_n[x] e^{j2\pi \frac{xm}{N}} = N \cdot IDFT(f_n[x]) \tag{2.18}$$

and with $N = f_{sample}/\Delta_f$ and

$$f_n = [0\, C_{n,1}\, C_{n,2}\, \ldots\, C_{n,26}\, 0\, 0\, 0\, 0\, 0\, 0\, 0\, 0\, 0\, 0\, 0\, C_{n,-26}\, \ldots\, C_{n,-2}\, C_{n,-1}] \tag{2.19}$$

Note that in HiperLAN/2 $N = 64$. Because N is a power of two, the efficient IFFT algorithm can be used to calculate the complex base band samples.

Applying the IFFT algorithm on the complex subcarrier symbols (see equation 2.18) results in 64 complex OFDM samples. The cyclic prefix can be generated by copying the last 16 complex time samples of the useful data part of an OFDM symbol and transmit them before transmitting the regular 64 samples. This and some timing definitions will be discussed in the following section.

2.7.3 Cyclic prefix

Each data OFDM symbol is proceeded by a *cyclic prefix*. This is an exact copy of the last T_{CP} seconds of the signal that represents the current OFDM symbol. In figure 2.5 time definitions, that we will use in this report, are outlined. We define useful data part duration T_U and the total symbol duration T_S.

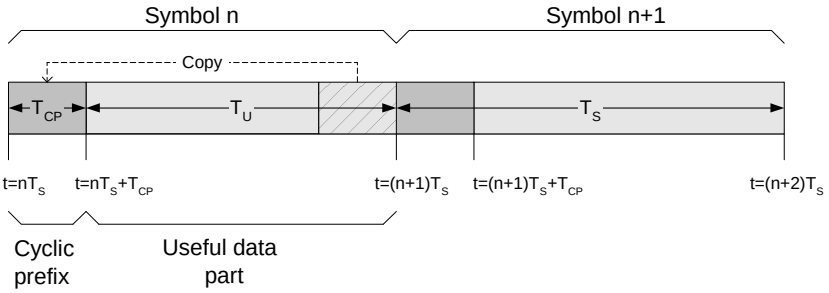


Figure 2.5: Timing definitions

In the HiperLAN/2 standard of the physical layer, two alternative durations of the cyclic prefix are defined: $T_{CP} = 0.8\ \mu s$ and $T_{CP} = 0.4\ \mu s$. The later is defined as optional to the transmitter and the receiver. In this report we will only work with $T_{CP} = 0.8\ \mu s$. Table 2.14 gives other durations used in HiperLAN/2.

2.8 Physical burst generation

A train of OFDM symbols containing data of higher protocol layers is proceeded by a *preamble*. This preamble consists of special OFDM symbols, that are known to the receiver. The configuration of the preamble dependents on the *burst type*. In this report will not be explained when and why certain burst types are used. The five burst types in HiperLAN/2 are (see [8]):

- Broadcast burst
- Downlink burst
- Uplink burst with short preamble
- Uplink burst with long preamble
- Directlink burst

All burst types use one or more so called *preamble sections* to precede the data burst. The standard of the HiperLAN/2 physical layer defines the preamble sections by their subcarrier values and duration. Before transmission of a section, OFDM is applied to the subcarrier values (see section 2.7). The preamble sections are depicted in figure 2.6. The choice of subcarrier values makes that the preamble sections consist of *repetitions* of identical parts. This is demonstrated for preamble section A in figure 2.7.

Preamble section A consists of five parts of 16 samples, denoted in [8] as A IA A IA IA[7]. Applying OFDM to the subcarrier values results in four parts, namely A IA A IA and hence the last necessary IA can be seen as a postfix to the section. Preamble section A has a duration of 4.0 μs. Before transmission of the preamble section the complex time samples are multiplied with $\sqrt{13/6}$. Note that preamble section A in fact uses QPSK loaded subcarriers.

There are two types of preamble section B (see figure 2.6 b and c). Both use a special case of QPSK loaded subcarriers, namely the distance between the subcarrier values is as large as possible. The short version of preamble section B consists of five parts of 16 samples: B B B B IB (see [8]) and has a duration of 4.0 μs. The first four parts are generated by applying OFDM to the subcarrier values. The last part IB is a postfix to the section, that is generated by inverting the sign of the last 16 proceeding samples. The long version of preamble section B is represented by ten parts: B B B B B B B B IB and has a duration of 8.0 μs. Before transmission the complex time samples are multiplied with $\sqrt{13/6}$.

[7] I in front of a letter means a *sign inversion* for all samples in that preamble section part.

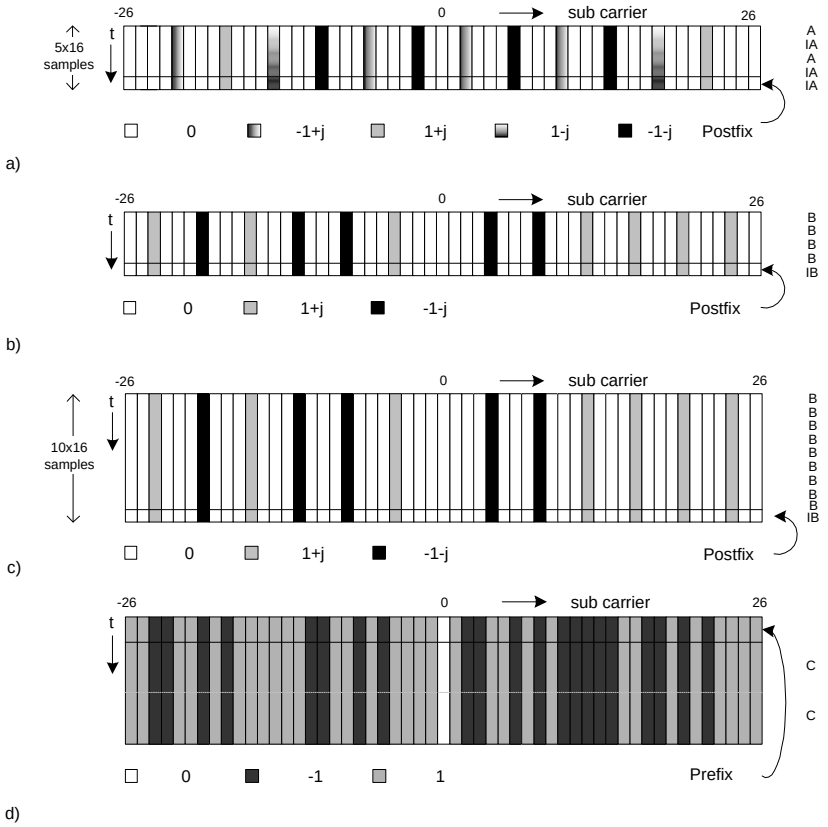


Figure 2.6: Preamble sections used to create a preamble to the data OFDM symbols. a) Preamble section A, b) Preamble section B (short), c) Preamble section B (long) and d) Preamble section C. The horizontal axis represents the subcarrier index l. The shade represents the value of the subcarrier. The preambles sections in a) and b) have a duration of 4.0 μs and in c) and d) 8.0 μs.

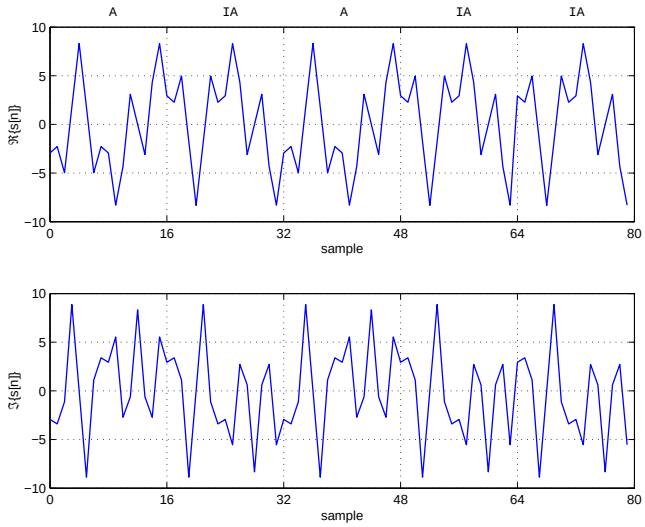


Figure 2.7: The choice of subcarrier values in preamble section A makes that the preamble consist of (sign inverted) repetitions of 16 samples

Preamble section C (see figure 2.6 d) has a duration of 8.0 μs and uses in fact BPSK modulated subcarrier values. OFDM creates 64 complex time samples of the 52 subcarriers values. After a prefix –the last 32 samples of the preamble section OFDM symbol–, the symbol is transmitted twice.

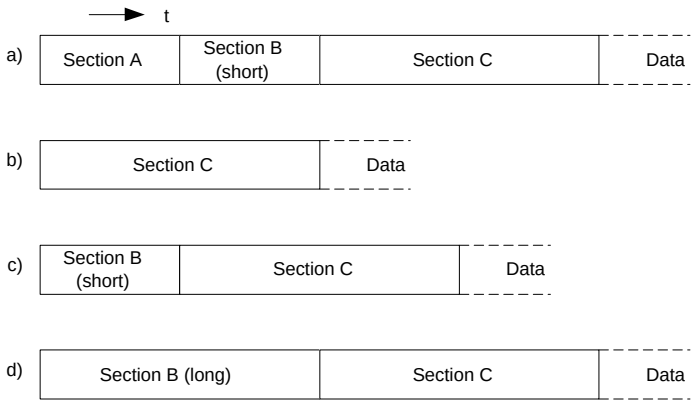


Figure 2.8: Preamble structure. a) Broadcast burst, b) downlink burst, c) uplink burst with short preamble and d) uplink burst with long preamble or directlink burst

Figure 2.8 shows which preamble structure is used with a certain burst type.

The preamble sections and the burst types have been implemented in the transmitter model.

2.9 Transmission of the burst

In this section the radio transmission of the burst will be discussed. First we will have a closer look at the spectrum of the baseband signal. Next the *carrier frequency allocation* will be discussed shortly. The complex baseband signal is converted to a *bandpass* signal. This is described in section 2.9.3.

The ETSI HiperLAN/2 physical layer standard [8] defines *transmit masks*; demands for the transmitted power in frequency bands other than the intended frequency band. A HiperLAN/2 transmitter should comply with these masks, since they limit the distortion, that transmitter may cause to other systems and neighboring HiperLAN/2 channels.

2.9.1 Spectrum of baseband signal

The complex envelope $\tilde{s}(t)$ of the transmitted bandpass signal is given by (see section 2.7 and [8]):

$$\tilde{s}(t) = \sum_{l=-\frac{N_{ST}}{2}}^{\frac{N_{ST}}{2}} C_l \; e^{-\jmath 2\pi l \Delta_f t} \tag{2.20}$$

The amplitude spectrum is:

$$\tilde{S}(f) = \sum_{l=-\frac{N_{ST}}{2}}^{\frac{N_{ST}}{2}} C_l \; \delta(f - \Delta_f l) \tag{2.21}$$

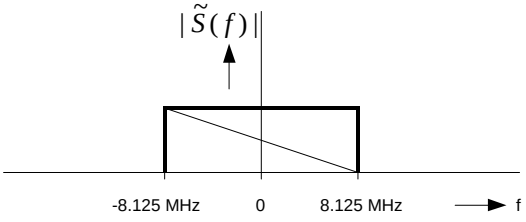


Figure 2.9: Spectrum of $\tilde{s}(t)$

This frequency spectrum is calculated using the Fourier property (see [10]):

$$g(t)e^{\jmath 2\pi f_\zeta t} \longleftrightarrow G(f - f_\zeta) \tag{2.22}$$

So the entire frequency spectrum of $g(t)$ is shifted and the DC value is now located at $f = f_\zeta$ instead of $f = 0$. Note that there is no mirrored spectrum at $f = -f_\zeta$, as would be the case for real signals.

In equation 2.21 the relation between the subcarrier number and the frequency spectrum becomes clear.

2.9.2 Carrier frequency allocation

The HiperLAN/2 carrier frequencies are located in two frequency bands: from 5.150 GHz to 5.350 GHz and from 5.470 GHz to 5.725 GHz (see [8]). The HiperLAN/2 carrier frequencies are spaced 20 MHz apart. These 20 MHz bands are called *channels*.

In this report we will use f_c to denote the carrier frequency that should be used for the transmission, f_s for the carrier frequency that is actually used for the transmission by the transmitter and f_r for the down-mixed frequency in the receiver. These definitions will be used in chapter 3.

2.9.3 Bandpass signal

The complex envelope $\tilde{s}(t)$ is transformed to the *bandpass signal* $s(t)$, centered at the carrier frequency f_c, with the following equation (see [8]):

$$s(t) = \sqrt{2} \cdot \Re\{\tilde{s}(t)e^{j2\pi f_c t}\} \tag{2.23}$$

Note that $s(t)$ is a *real* signal and hence will have a two sided, symmetrical frequency spectrum (see [10]).

Assume that the carrier frequency is much larger than half the bandwidth of the modulated signal:

$$f_c \gg \Delta_f \frac{N_{ST}}{2} \tag{2.24}$$

For HiperLAN/2 signals this assumption will be correct, since transmission takes place in the 5 GHz band ($f_c \approx 5 \cdot 10^9$ Hz) and half the width of the baseband spectrum is approximately $8\,10^6$ Hz.

Using the Fourier property (see [10]):

$$\Re\{g(t)\} \longleftrightarrow \frac{1}{2}[G(f) + G^*(-f)] \tag{2.25}$$

we can calculate the frequency spectrum of the bandpass signal $S(f)$:

$$S(f) = \frac{1}{\sqrt{2}} \cdot [\tilde{S}(f - f_c) + \tilde{S}^*(-f + f_c)] \tag{2.26}$$

Note that this signal has a component centered at f_c and a component centered at $-f_c$ (see figure 2.10)

2.10 Expected performance for AWGN channel

In an additive white Gaussian noise -*AWGN*- channel, the following relation exists for a BPSK modulated system (see [14]):

$$p_e = Q\left(\sqrt{2\,\frac{E_b}{N_0}}\right) \tag{2.27}$$

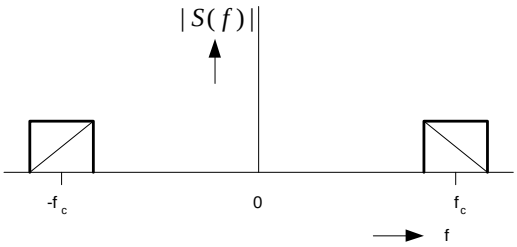


Figure 2.10: Spectrum of the real signal $s(t)$

with

$$Q(z) \triangleq \frac{1}{\sqrt{2\pi}} \int_z^\infty e^{-\frac{\lambda^2}{2}} d\lambda \tag{2.28}$$

and p_e the raw bit error probability.

If the OFDM system is compared to a "regular" BPSK system, we note that the OFDM system uses 48 BPSK channels in parallel and that the system "wastes" 1/5 of the transmitted power to the cyclic prefix. So the total signal power P_{s_OBPSK} for an OFDM system using BPSK compared to the P_{s_BPSK} of a BPSK system writes:

$$P_{s\_OBPSK} = \frac{5}{4} \cdot (48 \cdot P_{s\_BPSK} + 4 \cdot P_{pilot}) \tag{2.29}$$

The four pilot carriers are modulated BPSK, with the same power as the information bearing subcarriers. Thus the equation above evaluates to:

$$P_{s\_OBPSK} = 65 \cdot P_{s\_BPSK} \tag{2.30}$$

The noise power during the cyclic prefix in the OFDM symbol has no effect on the bit error rate of the system. Hence the noise power is scaled by 4/5.

The bit-rate of the OFDM system is higher than of the BPSK system:

$$R_{OBPSK} = \frac{4}{5} \cdot 48 \cdot R_{BPSK} \tag{2.31}$$

The resulting expression for the average bit energy of OFDM is:

$$E_{b\_OBPSK} = \frac{P_{s\_OBPSK}}{R_{OBPSK}} = \frac{65 \cdot P_{s\_BPSK}}{38.4 \cdot R_{BPSK}} \approx 1.69 \cdot E_{b\_BPSK} \tag{2.32}$$

Thus

$$\frac{E_{b\_OBPSK}}{N_{0\_OBPSK}} = \frac{65}{48} \frac{E_{b\_BPSK}}{N_{0\_BPSK}} \tag{2.33}$$

From this can be concluded that the OFDM system using BPSK as subcarrier modulation techniques needs an extra ≈ 1.3 dB to reach the same raw bit error rate as a "regular" BPSK system. This calculation is valid for all other subcarrier mapping techniques, because the average subcarrier energy is kept

equal to that of the BPSK mapping and equal to the pilot carrier energy. In [9] and [15] is the BER E_b/N_0 relationship described for BPSK, QPSK and QAM systems. In the next paragraph those relationships will be rewritten with the finding for HiperLAN/2 OFDM.

The theoretical BER as function of E_b/N_0 for a HiperLAN/2 QPSK system is given by:

$$p_{e\_OQPSK} = Q\left(\sqrt{\frac{48}{65}\frac{E_b}{N_0}}\right) \tag{2.34}$$

and for 16QAM:

$$p_{e\_O16QAM} = \frac{1}{4}\left(1 - \frac{1}{\sqrt{16}}\right) Q\left(\sqrt{\frac{48 \cdot 3 \cdot 4}{65 \cdot (2 \cdot 16 - 1)}\frac{Eb}{N0}}\right) \tag{2.35}$$

and for 64QAM:

$$p_{e\_O64QAM} = \frac{1}{6}\left(1 - \frac{1}{\sqrt{64}}\right) Q\left(\sqrt{\frac{48 \cdot 3 \cdot 6}{65 \cdot (2 \cdot 64 - 1)}\frac{Eb}{N0}}\right) \tag{2.36}$$

Figure 2.11 shows the expected raw bit error rate for the HiperLAN/2 system and figure 2.12 shows the expected bit error rate after error correction in the receiver.

In section 2.2.3 we calculated that the minimum sensitivity at a packet error rate of 10% translates to a bit error probability of $P_b = 2.4\,10^{-3}$. Table 2.15 summarizes the theoretical minimum E_b/N_0 requirements for the different bit-rate modes of the HiperLAN/2 system (see also figure 2.12).

Table 2.15: Minimum theoretical E_b/N_0 requirements to reach a PER of 10% using packet length of 54 bytes. The values are determined using figure 2.12

Bit-rate mode	Sub carrier modulation	R_c	Minimum E_b/N_0 [dB]
A	BPSK	1/2	3.0
B	BPSK	3/4	5.0
C	QPSK	1/2	6.0
D	QPSK	3/4	8.0
E	16QAM	9/16	11.0
F	16QAM	3/4	12.0
G	64QAM	3/4	15.0

2.11 Transmitter model implementation

The following functions are implemented in the HiperLAN/2 physical layer of the transmitter simulation model:

- Mapping

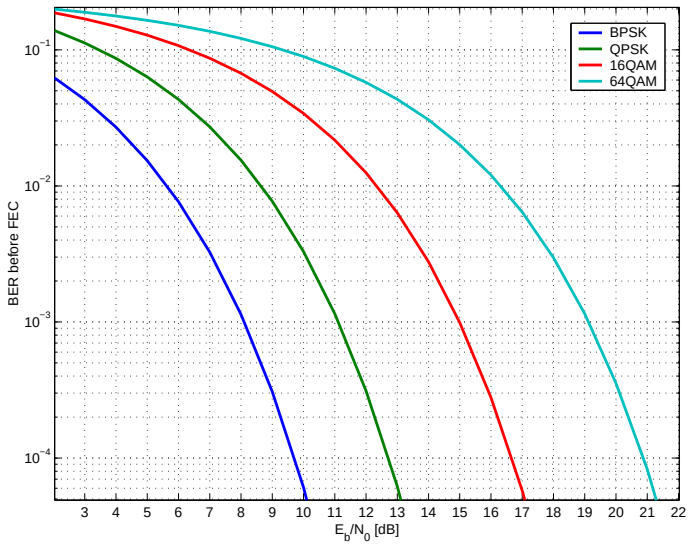


Figure 2.11: Expected BER of HiperLAN/2 before error correction

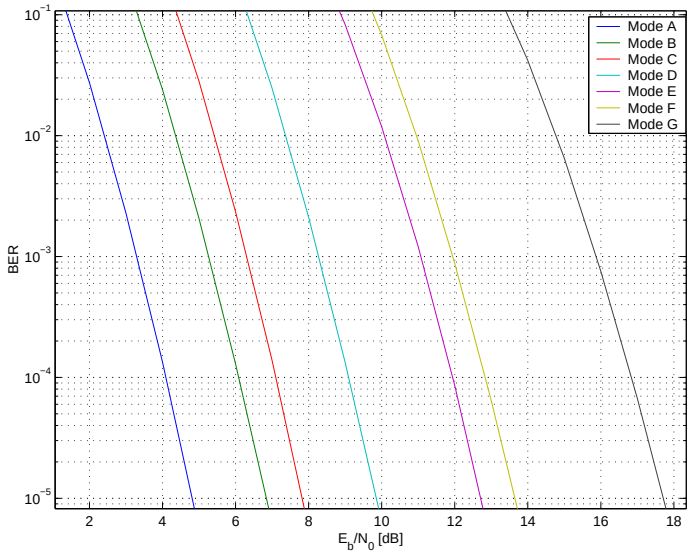


Figure 2.12: Expected BER of HiperLAN/2 after error correction (see table 2.3)

- Orthogonal frequency division multiplexing
- Physical burst generation

The C++ source code of the model is printed in a separate document [7]. Section A.1 of that document explains the choice of the *simulation software* that is used. In sections A.2 and sections A.3 the general model structure and common functions are presented. The source code of the transmitter is printed in sections A.4.1 to A.4.6 of the document [7].

The testing of the transmitter simulation model implementation is discussed in [16]; the transmitter model passed its functional test.

2.12 Conclusion

In this chapter the functionality of the physical layer of HiperLAN/2 is discussed. This is done following the concept configuration as proposed in [8].

Besides insight in the HiperLAN/2 system, this chapter provides a model of a HiperLAN/2 transmitter and theoretical AWGN performance of the system is deduced.

The following conclusions are drawn in this chapter:

- Input bits of the physical layer are scrambled. This causes a possible improvement of the BER.
- A convolutional encoder in combination with bit-rate mode dependent puncturing is used to apply FEC coding to the scrambled bits. The performance of the FEC coding in combination with an ideal decoder was analytical determined. The result is shown in figure 2.4.
- The FEC coded bits are interleaved to transmit neighboring bits on different subcarriers. This improves the BER in fading channel environments.
- The bits are mapped using BPSK, QPSK, 16QAM or 64QAM. The average power of the transmitted signal is kept equal for all modulation types.
- The system uses four pilot carriers (to the receiver known values).
- OFDM is used as modulation technique.
- A cyclic prefix is added to the signal.
- The theoretical performance of the HiperLAN/2 OFDM system on an AWGN channel was outlined in section 2.10. The results are depicted in figures 2.11 and 2.12 and table 2.15. These results can be used to test a decoder.
- A HiperLAN/2 physical layer simulation model of the transmitter has been implemented. It models the mapping, OFDM and physical burst generation functions. It passed its functional test.

Chapter 3

Signal Distortions in the HiperLAN/2 System

3.1 Introduction

The previous chapter described how the physical layer of the HiperLAN/2 system creates a bandpass signal that contains information about the bit stream at the input of the layer. This chapter discusses disturbances of this bandpass signal, that are caused by the indoor radio channel and the receiver hardware. figure 3.1 shows the configuration of the communication system of the SDR project (see [17]).

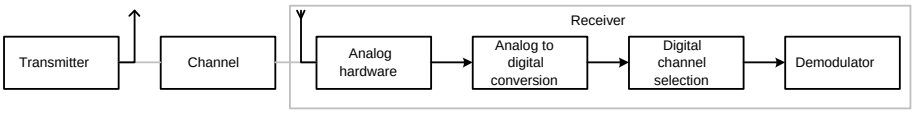


Figure 3.1: Overview of the SDR projects system configuration (see [17])

The channel guides the modulated signals from the transmitter to the receiver. In most cases the HiperLAN/2 connections will be situated in an indoor environment. Although the HiperLAN/2 connection is wireless, the terminals will only move at a slow speed for instance at walking speed. In section 3.2 the propagation mechanisms of an indoor radio channel are discussed. After the explanation of the propagation mechanisms, we will discuss the Rayleigh channel model; a model that seems reasonable for an indoor environment. The findings in section 3.2 will be useful for designing a channel equalizer, discussed in chapter 4.

The analog hardware of the receiver can cause some disturbances to the signal. In section 3.3 the analog hardware used in the SDR project is presented and *frequency offset*, *phase offset* and *phase noise* are discussed.

After the analog hardware, the HiperLAN/2 signal is sampled. Sampling causes *quantization noise* in the signal.

In chapter 2 we have seen that the HiperLAN/2 signal is transmitted in bands of 20 MHz. The receiver must select one of those bands and filter out unwanted signal components, before demodulation can take place. The channel selection filter causes a distortion in the band of interest. This will be discussed in section 3.5.

Another distortion of the signals in the receiver is the fact that numbers are represented with a *finite* number precision (see [18] and [19]). When operations like "+" and "/" are carried out, rounding or truncating of signal values takes place. This introduces noise in the signals. In section 3.6 this distortion will be discussed and an algorithm will be presented, that can be used to simulate a certain number precision.

Throughout this chapter we will prove, that all these disturbances can be simulated at *baseband level*[1] , instead of the real world *bandpass* signal. This is convenient for simulation of the system, because it reduces the *execution time* of the simulation, where the simulated time remains equal. Note that a simulation system should work with simulation time steps, that represent –at least– two times the highest frequency of the process to be simulated.

3.2 Indoor radio channel

HiperLAN/2 systems will be used in a wide range of environments, like offices, large buildings (i.e. stock exchanges or exhibition halls) and residential environments (see [20]). Five different channel models have been made by ETSI, to be able to simulate all these environments. The channel models are described in [21]. In [20] simulation results are presented using the channel models. In the first stage of the SDR project, no actual transmitting will take place through the indoor radio channel and hence the channel models will not be used in this report. A future study should implement the channel models.

Three physical propagation mechanisms play a role in an indoor radio channel (see [22], [23], [24] and [25]):

- Reflection
- Diffraction
- Scattering

These mechanisms are all described by the Maxwell equations. The following sections explain these effects and how these effects can be modelled.

[1]The spectrum of the signal is centered around $f = 0$. *Bandpass* means that the spectrum is centered around another frequency: $f \neq 0$.

3.2.1 Multipath propagation mechanisms

Reflection

A radio wave is (partially) reflected by an object with a smooth surface that is large compared with the wavelength of the signal. A part of the signal energy is absorbed by the material. So both reflected and transmitted signal are attenuated (see figure 3.2). In the frequency range our system of interest operates, the transmitted signal is only moderately attenuated. In an office environment walls, floors and furniture are the main contributors to this effect.

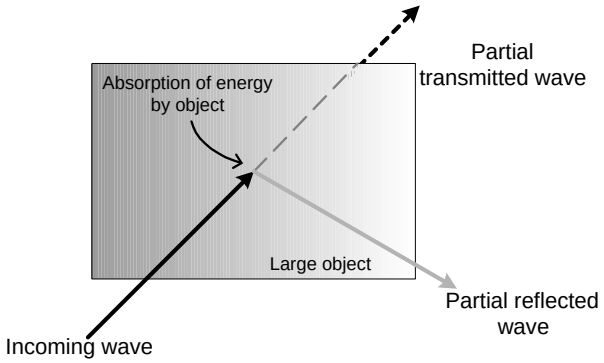


Figure 3.2: Reflection of waves on large objects compared to wavelength

Scattering

Scattering occurs when the radio waves incident on an object which size is about or smaller the signal wavelength (see figure 3.3). The energy of the radio wave will scatter in many directions. This phenomenon is cased by in example metallic studs or cabinets.

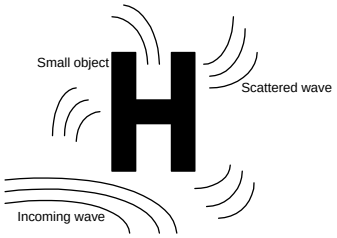


Figure 3.3: Scattering of wave on objects which size is about or smaller the wavelength

Diffraction

A part of the radio wave bends into the shaded area behind a large object even if the object is impenetrable to the radio wave (see figure 3.4). This effect is caused by edges of walls, windows and other large objects. The

energy contributions of diffracted paths can be large. The signal variations can be as large as 20 dB in some situations.

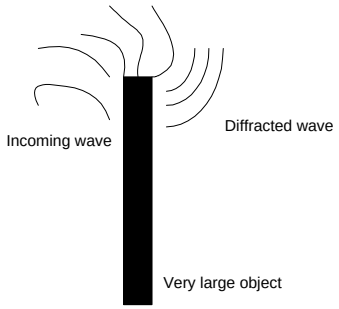


Figure 3.4: Diffraction of wave on very large objects compared to wavelength

All these mechanisms cause delayed and attenuated versions of the transmitted signal to arrive at the receiver (see [25]). Several models have been made to describe these effects. The Rayleigh channel model is a suitable model to describe multipath propagation in an indoor office environment (see [25], [22] and [26]). Note that this model does not cope with non-linear distortions.

3.2.2 Rayleigh Channel Model

As discussed above, the modulated signal is transmitted through an air interface. This implies that there are several ways for the transmitted signal to reach the receiver. Walls, furniture and all other objects that may be located in a reasonable distance from transmitter and receiver reflect the electromagnetic waves. All these signals may travel a different distance before they reach the receiver. This is called multipath transmission. A different distance means that some signals are delayed more than others are. Even different frequency components in "one" path may be delay different than others in the same path. All the signals are combined in the antenna and they form the received signal.

This type of channel is often modelled by the Rayleigh channel model. The received signal is represented by:

$$r(t) = \sum_{i=0}^{N_P-1} \beta_i s(t - \tau_i) \tag{3.1}$$

with:

i – path index

N_P – total number of paths

$s(t)$ – transmitted signal[2]

[2] $s(t)$ and $r(t)$ are *bandpass* signals.

$r(t)$ – received signal

τ_i – delay for path i

β_i – complex gain of path i

All of the possible infinite number of radio waves are delayed a certain time before they reach the receiver (see figure 3.5). It can be proven that, when the transmitted signal is a pure sinusoid with no noise and no initial phase, so

$$s(t) = cos(\omega_c t) \tag{3.2}$$

with:

ω_c – frequency of transmitted signal (radians/s),

the received signal $r(t)$ is given by:

$$r(t) = A(t)cos(\omega_c t + \theta(t)) \tag{3.3}$$

With amplitude $A(t)$ is Rayleigh distributed and the phase $\theta(t)$ is uniform distributed (see [22]).

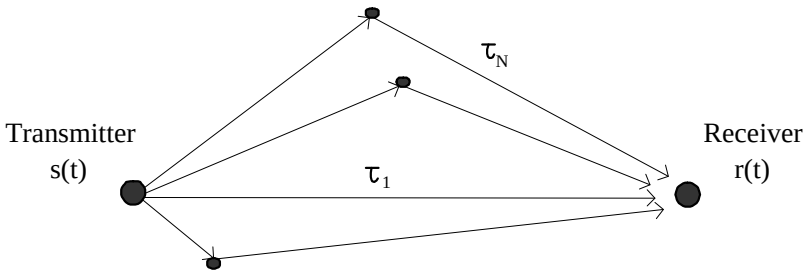


Figure 3.5: Rayleigh channel model

3.2.3 Delay spread and coherence bandwidth

In the previous section we have seen that propagation mechanisms cause the receiver to receive multiple delayed and attenuated versions of the transmitted signal. There are several ways to describe the multipath delay properties[3] of channels. The basis of an often found the description in literature, is the so called *power delay profile*, the power of the received delayed signals is plotted as function of its *excess delay*. The first version of the transmitted signal to arrive at the receiver is said to have a excess delay of zero. So all delay times are compared to the first arriving component.

The standard deviation σ_τ of the delay of all paths is called *RMS delay spread* and is a parameter that describes the type of *fading* in the channel. It

[3]The statistical properties of β_i and τ_i.

is important to make the difference between flat fading and frequency selective fading clear. Flat fading means that all spectral components are affect by the channel in a similar matter (see [22] and [23]).

Coherence bandwidth W_C is defined as (see [22] and [23]):

$$W_C = \frac{1}{2\pi \cdot \sigma_\tau} \tag{3.4}$$

It gives rise to a frequency range over which the propagation varies about 3 dB and remains nearly flat.

When the delay spread is smaller than one tenth of the symbol duration or when the bandwidth W of the transmitted signal is much smaller than the coherence bandwidth:

$$\sigma_\tau < \frac{1}{10} \cdot T_S \tag{3.5}$$

$$W \ll W_C \tag{3.6}$$

a channel is said to exhibit flat fading, otherwise it is called frequency selective fading (see [22] and [23]).

In table 3.1 some typical RMS delay spreads are given. From this table can be concluded that the HiperLAN/2 OFDM system exhibits flat fading.

Table 3.1: Typical delay spreads (frequency range: 4 GHz – 6 GHz). These values are found in [23].

Environment	Maximum delay [ns]
Large building (i.e. Stock exchange)	120
Factory	125
Office building	130
Single office room	30

In section 2.7.3 we discussed the transmission of a cyclic prefix, before the actual useful part of the symbol is transmitted. The cyclic prefix prevents that attenuated and delayed versions of the previous signal interfere with the current symbol. Note that the duration of the prefix is 800 ns, while the maximum delay in an office building is 130 ns (see table 3.1).

3.2.4 Doppler shift and coherence time

Doppler frequency shift happens when radio waves bounce or scatter of moving object or when the terminals are moving. In other words, Doppler shift causes the path delays τ_i to become time dependent. In an indoor office environment movements are generally very slow compared to the data rate and for practical purposes the channel usually can be regarded as stationary.

Doppler shift f_m is given by:

$$f_m = \frac{\lambda}{v} \tag{3.7}$$

where λ is the wavelength and v the speed of the object, either transmitter, receiver or reflecting object.

The maximum expected Doppler shift in an indoor office environment for a system transmitting at 5 GHz and a maximum velocity of $v = 2.5\ m/s$ is $f_m = 42$ Hz (about 1 millionth % of the transmission frequency). This is probably a small value compared by frequency drifts of oscillators in transmitter and receiver (see section 3.3.1). The effect that Doppler shift causes, is called *frequency dispersion*.

Frequency dispersion can be translated to *coherence time*. This specifying measure describes in what time period received signals have a strong correlation in their amplitude. This is the time over which the channel can be seen time invariant. There is no exact relation between the maximum Doppler shift and coherence time, but [27] uses as approximate relationship:

$$T_C = \frac{0.423}{f_m} \tag{3.8}$$

When we assume that $f_m = 42$ Hz, the coherence time equals $T_C = 10.1\ ms$. Thus the channel equalizer in a HiperLAN/2 receiver should be *adaptive* and it should update at least once per 10.1 ms. Note that this time is long compared to a HiperLAN/2 transmission burst of 2 ms (see [20] and [6]).

3.2.5 Transfer function

The above described statistical channel properties can be described with a complex transfer function. For each delay time τ there is a different attenuation and phase shift. Due to motion of objects the coefficients of the transfer function change over time. The channels impulse response follows directly from equation 3.1:

$$h(t,\tau) = \sum_{i=0}^{N-1} \beta_i(t)\ \delta\left(\, t - \tau_i(t) \,\right) \tag{3.9}$$

In [22] a simulation of such transfer function is given. It makes clear how the filter coefficients can be calculated. The updating of the coefficients is defined by the coherence time T_C (see section 3.2.4).

3.2.6 Baseband description of the indoor radio channel

In this section we will find that the above discussed channel can be described at baseband level. This has two advantages:

- The channel can be modelled at baseband level. This reduces the execution time of a simulation, as discussed in the introduction of this chapter.
- The channel equalizer in the receiver can operate at baseband level, since the real channel can be described at baseband level.

The received signal is given by (noise and interference are discussed in section 3.2.7 and will be omitted here):

$$R(f) = S(f)H(f) \tag{3.10}$$

Where $H(f)$ is the transfer function of the channel.[4]

At the receiver we need to convert the bandpass signal $r(t)$ to its complex envelope $\tilde{r}(t)$, in order to decode the subcarrier symbols C_l (see equation 2.20). In other words, the signal $\tilde{r}(t)$ needs to be shifted in frequency domain with $\pm f_r$.[5] We use an intermediate variable $u(t)$ to investigate the effect of a frequency shift:

$$u(t) = r(t)e^{-\jmath 2\pi f_r t} \tag{3.11}$$

Thus

$$U(f) = R(f + f_r) \tag{3.12}$$

Which equals, using equation 3.10 and 2.26:

$$U(f) = \frac{1}{\sqrt{2}} \cdot [\tilde{S}(f) + \tilde{S}^*(-f + 2f_r)] \cdot H(f + f_r) \tag{3.13}$$

Figure 3.6: Spectrum of $u(t)$ (see equation 3.13). A channel transfer function $H(f) = 1 \; \forall \; f$ is used for simplicity

The resulting complex spectrum of $u(t)$ is plotted in figure 3.6. The figure shows that the spectrum of the intermediate function $u(t)$ has two parts: one part centered at $f = 0$, and the other at $f = 2f_r$, while the transmitted baseband signal $\tilde{s}(t)$ has only one component at $f = 0$. To get rid of the component centered at $f = 2f_r$ an *ideal lowpass filter* can be used[6]:

$$H_r(f) = \begin{cases} 1 & -f_\kappa < f < f_\kappa \\ 0 & \text{otherwise} \end{cases} \tag{3.14}$$

With

$$\frac{N_{ST}}{2}\Delta_f < f_\kappa < 2f_c - \frac{N_{ST}}{2}\Delta_f \tag{3.15}$$

When a gain of $\sqrt{2}$ is applied, $\tilde{R}(f)$ can be written as:

$$\tilde{R}(f) = \sqrt{2} \cdot U(f) \cdot H_r(f) \tag{3.16}$$

[4]In this section we omit the fact that $H(f)$ is dependent on τ. However, the analysis remains valid for $H(f, \tau)$.

[5]In this section is assumed that $f_r = f_s = f_c$.

[6]This component will usually be dissipated in the receiver hardware.

Or

$$\tilde{R}(f) = \tilde{S}(f) \cdot \tilde{H}(f) \tag{3.17}$$

With $\tilde{H}(f)$ the baseband equivalent of the transfer functions:

$$\tilde{H}(f) = H(f + f_c) \cdot H_r(f) \tag{3.18}$$

From the above (equations 3.17 and 3.18) can be concluded that the system can be simulated at *baseband* using equation 3.18 as the transfer function of the channel. This means also that the receivers channel equalization may operate at baseband level.

3.2.7 Noise and interference

Our channel model is not yet complete. Other effects we have not taken in account yet are noise $\eta(t)$ and interference $i(t)$ of devices transmitting in the same frequency range as our system. The complete channel model is given in in figure 3.7.

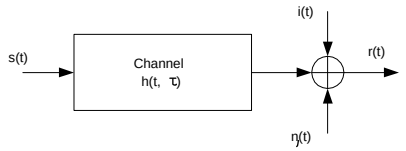


Figure 3.7: Channel model with noise and interference

Noise is caused by a few phenomena [26]:

- Thermal noise in electrical components for example in the front-end of the receiver
- Shot noise processes developed in electronic devices
- Electromagnetic radiation from earth, sun and other cosmic sources

The noise caused by these phenomena is commonly modelled as *additive white Gaussian noise*. The Gaussian process $\eta(t)$ has a zero mean and a constant power spectral density of $N_0/2$ $[Watt/Hz]$ (in the frequency range we are interested in), which equals the variation of the random process (see [10] and [14]).

3.3 Analog hardware architecture of the SDR receiver

In figure 3.8 the analog hardware configuration of the software defined radio project (see [17] and [28]) is shown. The task of this hardware is to create

complex, time discrete samples, that represent the base band HiperLAN/2 signal in the radio band of interest. This task is done by filtering the antenna signal, amplifying the signal with a *low noise amplifier* (*LNA*), down mix the signal to baseband and to filter unwanted components from the signal. Finally the signal is sampled.

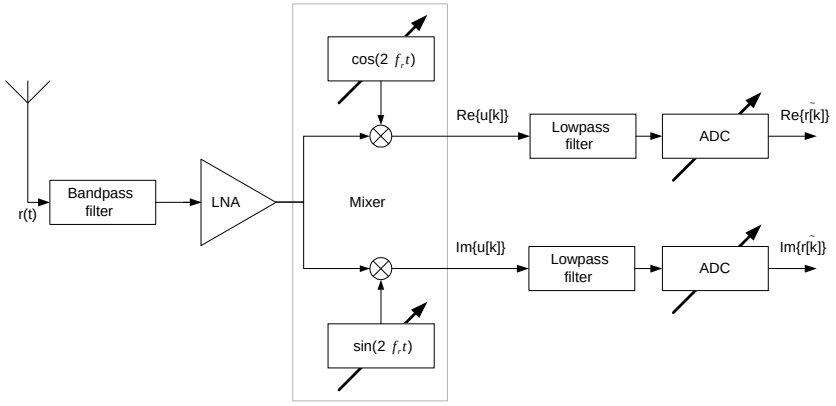


Figure 3.8: Software defined radio receiver hardware (see [17])

In the following sections some distortions caused by the analog hardware are discussed. The analog hardware is currently being developed, so at this point in time a *characterization* of the hardware is not yet available (see [28] for some preliminary results). In this report we will assume that the bandpass filter and the amplifier are ideal.[7] In other words, no non-linear distortions are expected, that can not be solved by the channel equalizer in the receiver (see section 4.10). The distortions that will be discussed in this report are: frequency offset, phase offset and phase noise.

The hardware of the receiver (see figure 3.8) is driven by a frequency source, usually a *voltage controlled oscillator* (*VCO*). At this moment in time, the characteristics of the reference frequency source of the receiver in the SDR project is not yet determined. In [31] some important parameters of VCO's are explained. Most important parameter to this and the following section is the *spectral purity* of the oscillator. It tells how close the frequency of the oscillator will be to its intended frequency and how stable this frequency is.

The oscillator determines the operation of the mixer that mixes the incoming signal to baseband and it determines the time-base of the receiver, in other words the *sampling frequency* of the analog to digital converter (*ADC*).

[7] OFDM has a high peak-to-average power ratio, hence a good linearity is expected of the amplifiers in the system (see [12], [23], [24], [29] and [30]).

3.3.1 Frequency offset

In this section we will determine the influence of a frequency offset in the mixer on the performance of the HiperLAN/2 system.

Assume that there is a difference between the carrier frequency in the transmitter and the receiver of f_Δ[8]. Then

$$f_r = f_s + f_\Delta \tag{3.19}$$

If we substitute this in equation 3.11, this results in:

$$u(t) = r(t)e^{-\jmath 2\pi(f_s+f_\Delta)t} \tag{3.20}$$

Since

$$r(t) = \sqrt{2} \cdot \Re\{\tilde{s}(t)e^{\jmath 2\pi f_s t}\} \star h(t) \tag{3.21}$$

we can write for $u(t)$ in the frequency spectrum:

$$U(f) = \frac{1}{\sqrt{2}} \cdot [\tilde{S}(f+f_\Delta) + \tilde{S^*}(-f+2(f_s+f_\Delta))] \cdot H(f+f_s+f_\Delta) \tag{3.22}$$

Thus, after applying the ideal lowpass filter and a gain of $\sqrt{2}$ (see section 3.2.6), the baseband signal $\widetilde{r}(t)$ evaluates to:

$$\widetilde{r}(t) = \left(\widetilde{s}(t)\, e^{-\jmath 2\pi f_\Delta t}\right) \star \widetilde{h}(t) \tag{3.23}$$

From the equation above can be concluded that a frequency offset can be described at baseband level as a time varying *rotation* of the transmitted complex baseband samples.

At this point we assume that the subcarrier values are retrieved from the sampled version of the baseband signal $\widetilde{r}(t)$ by applying a discrete Fourier transformation (DFT). We will also assume that the frequency offset f_Δ is smaller than the subcarrier spacing Δ_f (see section 2.7.2). A received subcarrier value $\hat{C}_l$ is given by:

$$\hat{C}_l = \frac{1}{N} \sum_{i=0}^{N-1} \widetilde{r}[i]\ e^{-\jmath 2\pi \frac{il}{N}} \tag{3.24}$$

If equation 3.23 is substituted in the equation above, this yields in[9]:

$$\hat{C}_l = \frac{1}{N} \sum_{i=0}^{N-1} \widetilde{s}[i]\, e^{-\jmath 2\pi \frac{f_\Delta}{f_{sample}} i}\ e^{-\jmath 2\pi \frac{l}{N} i} \tag{3.25}$$

The transmitted subcarrier value C_γ gives rise to the transmitted complex baseband signal[10] (see also equation 2.16):

$$\widetilde{s}[i] = C_\gamma\ e^{\jmath 2\pi \frac{\gamma}{N} i} \Big|_{\ i=0\,\ldots\,N-1} \tag{3.26}$$

[8]Note that this frequency difference is *not* time dependent. Any time dependent changes will be represented as *phase noise* (see section 3.3.2).

[9]For simplicity the assumption $h(t) = \delta(t)$ is made.

[10]We assume that only subcarrier γ is not equal to zero.

When equation 3.26 is substituted in equation 3.25 and the relation $f_{sample} = N\,\Delta_f$ is used, the result is:

$$\hat{C}_l = \frac{1}{N} \sum_{i=0}^{N-1} C_\gamma \; e^{\jmath 2\pi \frac{i}{N} \left(\gamma - l - \frac{f_\Delta}{\Delta_f}\right)} \tag{3.27}$$

Note that when no frequency offset is present ($f_\Delta = 0$), the received subcarrier value equals the transmitted subcarrier value $\hat{C}_l = C_\gamma$ only if $l = \gamma$.

From equation 3.27 can be concluded that a frequency offset causes *inter-subcarrier interference*. If a frequency offset is present of for example $f_\Delta = 1/10\ \Delta_f = 31.250$ KHz, and $C_\gamma = 1$, the received subcarrier $\hat{C}_{l=\gamma}$ will be slightly less than the transmitted subcarrier value and some of its energy will flow to neighboring subcarriers (see figure 3.9).

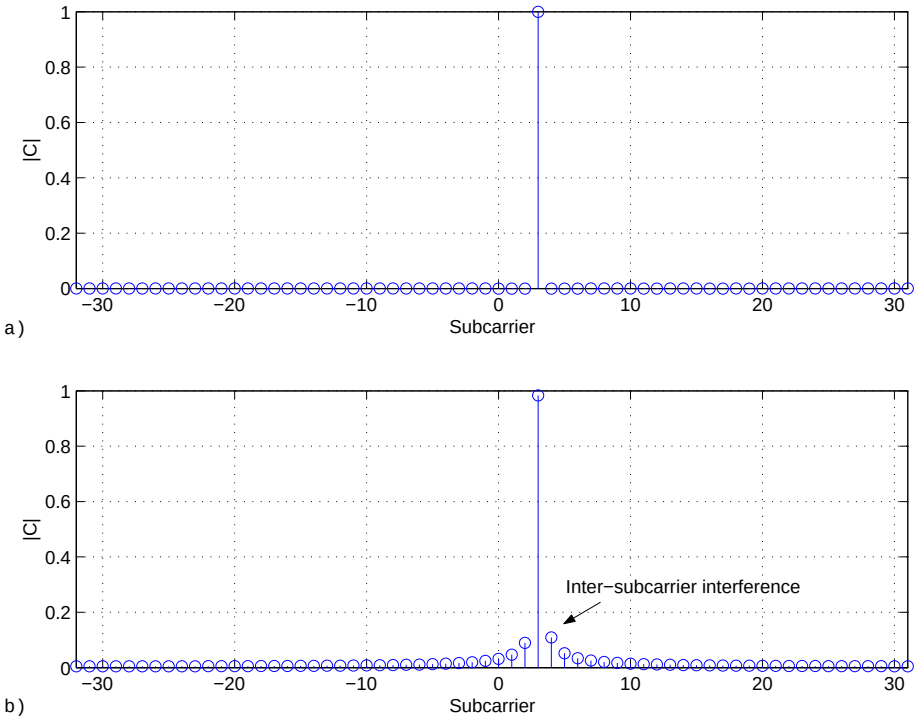


Figure 3.9: Influence of frequency offset on subcarriers; a) no frequency offset present and b) a frequency offset of $f_\Delta = 1/10\ \Delta_f = 31.250$ KHz. These results are calculated with equation 3.27.

The HiperLAN/2 standard [8] defines that:

$$\frac{f_c - f_s}{f_c} < 0.002\ \% \tag{3.28}$$

If we assume an equal demand for f_r compared to f_c, f_Δ may be up to 250 KHz. A quick calculation with equation 3.27 shows that at this frequency offset almost all power of a transmitted subcarrier is found in the neighboring subcarrier in

the receiver; large bit errors rates will occur. From this follows that the receiver should try to compensate for frequency offsets.

In figure 3.10 the power that spills into a neighboring subcarrier in the direction of the frequency offset is plotted. Note that the transmitted power on the subcarrier equals 1. This is calculated with equation 3.27 (see also [24] chapter 21).

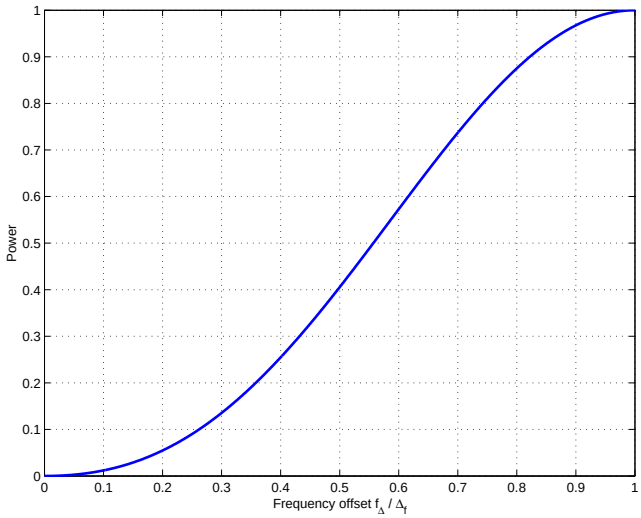


Figure 3.10: Influence of frequency offset on neighboring subcarrier in direction of the frequency offset

3.3.2 Phase offset and phase noise

The second type of distortions, that are caused by the process of down mixing the received bandpass signal to baseband, are phase disturbances in the down mix signal. In this section we will assume that there is *no* frequency offset present and that the mixers in the transmitter do not cause any phase disturbances.

Phase noise causes the performance of the system to decrease by two effects: loss of orthogonality of the subcarriers (analog to a *time dependent* frequency offset) and a *common phase offset* to all subcarriers. To describe these effects independently, we will divided the phase of the mixer compared to the mixer in the transmitter into two parts: a part that remains equal throughout an OFDM symbol, ψ_0 and a time varying part, $\psi(t)$.

For $u(t)$ we can write:

$$u(t) = r(t)e^{-\jmath 2\pi f_r t + \jmath\,\phi(t)} \tag{3.29}$$

with $\phi(t) = \psi_0 + \psi(t)$. First we will discuss the time independent part of the phase offset. Using a modified version of equation 3.23, the effect of the time

independent phase offset can be seen:

$$\widetilde{r}(t) = \left(\widetilde{s}(t)\, e^{\jmath\,\psi_0}\right) \star \widetilde{h}(t) \tag{3.30}$$

Assume that $h(t) = \delta(t)$. The effect of this phase rotation on the demodulated subcarrier values can be calculated using (modified) equations 3.25 to 3.27:

$$\hat{C}_\gamma = e^{\jmath\,\psi_0} C_\gamma \tag{3.31}$$

From this can be concluded that a time independent phase offset causes a phase rotation in the subcarrier values, that is equal to all subcarriers. This phase offset can result in a great reduction of the performance of the HiperLAN/2 system, because the entire constellation gets rotated, resulting in a high bit error rate. A receiver implementation should keep track of the common phase rotation and correct the distortion. The common phase rotation can be measured by the receiver by using *pilot* carriers (see section 2.7.1), since they are equally disturbed by the common phase offset.

The effect of the time dependent phase offset, so called *phase noise* or *phase jitter*, is much harder to describe, since it is a statistical process, that is dictated by the actual implementation of the mixer. Good descriptions are found in [24], [32] and [31]. In this report we will assume that $\psi(t)$ is the outcome of a Gaussian random process with mean zero and variation ${\sigma_\psi}^2$ (see also [24] section 19.6). We will use this only to proof that phase jitter causes inter-subcarrier interference.

Equation 3.30 (with $h(t) = \delta(t)$) can be rewritten for a time dependent phase offset (in sampled form) as:

$$\widetilde{r}[i] = \widetilde{s}[i]\, e^{\jmath\,\psi[i]} \tag{3.32}$$

This results in the following demodulated subcarrier values (see also equation 3.27):

$$\hat{C}_l = \frac{1}{N} \sum_{i=0}^{N-1} C_\gamma \; e^{\jmath\, 2\pi \frac{i}{N}(\gamma - l) + \jmath\,\psi[i]} \tag{3.33}$$

Figure 3.11 shows the inter-subcarrier interference that is caused by a phase jitter with ${\sigma_\psi}^2 = 0.01\ [rad^2]$. In the figure only one subcarrier value is transmitted. When these results are compared with the results of frequency offset (see section 3.3.1), we see that frequency offset introduces inter-subcarrier interference locally, while phase jitter causes inter-subcarrier interference in all subcarriers.

The inter-subcarrier interference can cause a great decrease in system performance. Hence in the designing of the receiver hardware, mixers with great spectral purity should be chosen.

A practical implementation of the above discussed *complex* mixer is that two mixers are used and in fact this implementation is drawn in figure 3.8. The phase difference between these mixers should be $\pi/2$. An error in this phase difference can be corrected with a channel equalizer (see [23] and [24]).

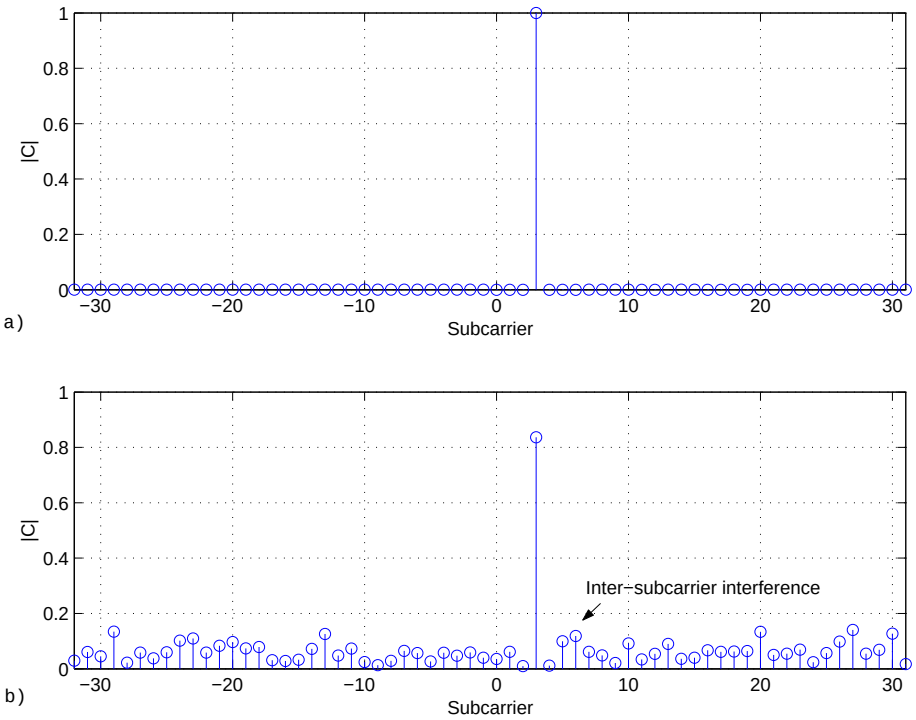


Figure 3.11: Influence of time dependent phase jitter on subcarriers; a) No phase noise present and b) Gaussian phase noise ($\sigma_\psi{}^2 = 0.01\ [rad^2]$). These results are calculated with equation 3.33.

3.4 Sampling the signal

Before the baseband signal is sampled, a lowpass filter is applied. This filter has two functions: *anti-aliasing filter* and coarse *channel selection filter*.

As initial design a 7^{th} order *Butterworth* filter has been designed (see [17]). The filter has a cut-off frequency of 10 MHz. Its measured transfer function is plotted in [28] and figure 3.12 shows the theoretical transfer function of the filter. From this figure can be concluded that the filter does have a influence on the subcarriers.

The lowpass filtered baseband signal is sampled at $f_{ADC} = 80$ MHz using a 12 bit analog to digital converter (see [17]). The ADC can be modelled as a *zero-order sample and hold*, followed by a *quantizer* and a signal *limiter* (see also [14] and [10]).

3.4.1 Symbol window drift

Because the sampling clock of the receiver hardware will not be synchronized with the timing in the transmitter, the symbol window in the receiver may slowly

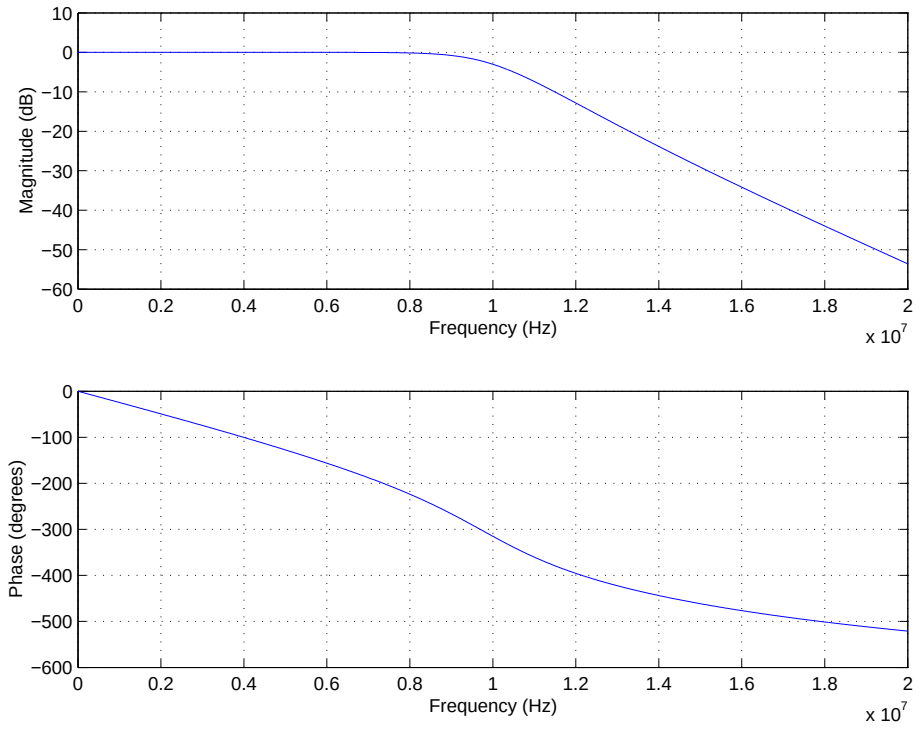


Figure 3.12: Theoretical transfer function of the anti-aliasing filter used in HiperLAN/2 receiver of SDR project (see [17] and [28])

wander away from the ideal symbol window, if the receiver does not adapt. A not perfect alignment, of the useful data part window –the *symbol window*– and the *observation window* used in the receiver, results in a large decrease of the performance of the system. In this section we will have a closer look at the effects of symbol window drift.

The HiperLAN/2 standard [8] defines that the transmitter should use one frequency source for carrier frequency f_s generation and clocking the time-base, f_{sample}. This frequency source should have a accuracy of $\pm$ 0.002 % (see equation 3.28). If we demand the same time-base accuracy for the receiver, this may lead to a drift of ≈ 1 sample per 300 OFDM symbols.

As explained in section 2.7.3, an OFDM symbol contains a cyclic prefix and a useful part. The ideal useful data part window of symbol n, is located between $nT_S + T_{CP} \leq t < (n+1)T_S$. We will denote the sample frequency in the receiver as f_{sample_R} and the sample frequency in the transmitter as f_{sample_T}.

If we assume a sampling clock offset:

$$\chi = \frac{f_{sample_T} - f_{sample_R}}{f_{sample_T}} \tag{3.34}$$

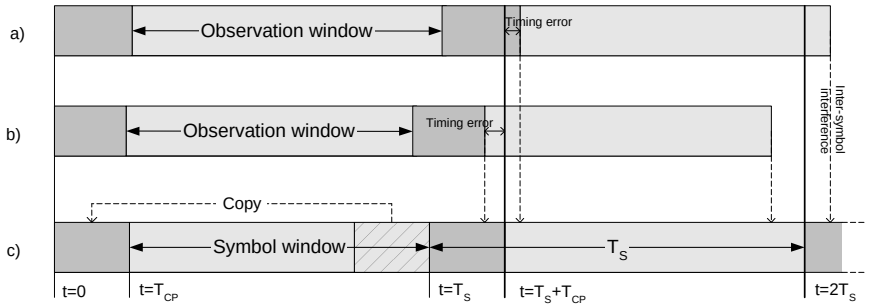


Figure 3.13: Definitions in the explanation of the effect of timing errors. a) $f_{sample_R} > f_{sample_T}$, b) $f_{sample_R} < f_{sample_T}$ and c) $f_{sample_R} = f_{sample_T}$. Figures a) and b) show the *expected* timing in the receiver, while c) shows the *actual* timing.

we can write for the time-base t' in the receiver[11]:

$$t' = (1 + \chi)t \tag{3.35}$$

The symbol window will appear to be drifting in the receiver, because, if the receiver does not adapt for the difference in sample frequencies, the symbol is expected to lie in:

$$(nT_s + T_{CP})(1 + \chi) \leq t < (n + 1)T_s(1 + \chi) \tag{3.36}$$

This results in a timing error of [12]:

$$t_{error} = (nT_s + T_{CP})\,\chi \tag{3.37}$$

Figure 3.13 c) shows the timing of the transmitter. This is the signal that is actually transmitted. Figures 3.13 a) and 3.13 b) show what the time-base in the receiver *expects* to have received. Note that in a.) *inter-symbol interference* occurs.

The effect of a timing error on the subcarrier values, can be determined by writing:

$$\widetilde{r}(t') = \widetilde{s}(t - t_{error}) \tag{3.38}$$

The sampled version of this signal is given by:

$$\widetilde{r}[i] = \widetilde{s}[i - i_{error}] \tag{3.39}$$

with

$$i_{error} = t_{error}\, f_{sample_R} \tag{3.40}$$

Applying DFT to equation 3.39 results in:

$$\hat{C}_l = \frac{1}{N} \sum_{i=0}^{N-1} C_\gamma \; e^{\jmath 2\pi \frac{i - i_{error}}{N} \gamma}\, e^{-\jmath 2\pi \frac{i}{N} l} \tag{3.41}$$

[11] Note that the time-base in the *transmitter* is assumed to be the correct one.

[12] In this analysis the fact that the observation window has a different length than the symbol window is omitted. This difference in length will be negligible compared to the error a timing error causes.

Rearranging of the powers of e makes the influence of a timing error on the subcarrier values clear:

$$\hat{C}_l = \frac{1}{N} \sum_{i=0}^{N-1} C_\gamma \; e^{-\jmath\, 2\pi \frac{i_{error}}{N} \gamma} \, e^{\jmath\, 2\pi \frac{i}{N} (\gamma - l)} \tag{3.42}$$

Thus

$$\hat{C}_\gamma = C_\gamma \, e^{-\jmath\, 2\pi \frac{i_{error}}{N} \gamma} \tag{3.43}$$

From the above can be concluded that each subcarrier value is *rotated* with an angle that is dependent on the subcarrier number. This analysis is also valid for timing error that are caused by *other reasons* than symbol window drift (for example *synchronization errors*). Note that a timing error of i_{error} samples means that the rotation over all subcarriers is in total $2\,\pi\, i_{error}$.

A timing error can cause a great decrease of the systems performance. For example, assume a timing error of one sample. In that case the outmost carriers ($l = -26$ or $l = 26$) will be rotated almost $\pm\pi$, which –of course– causes a large number of incorrectly received bits. A receiver implementation should be able to compensate for symbol window drift.

3.5 Digital channel selection

In [33] three digital filter designs are proposed to perform channel selection operations for HiperLAN/2. In this report we will not design a channel selection filter for the system, but we will merely try to find out what the *distortions* are caused by the filter. Besides the large reduction of signal components that can cause interference, a channel selection filter will also disturb the signal in the band of interest. The overall channel selection requirements for HiperLAN/2 are shown in figure 3.14 (see [2]).

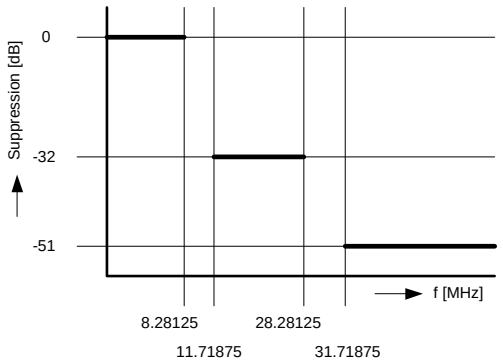


Figure 3.14: HiperLAN/2 overall channel selection requirements (see [8] and [2])

In this report we will only have a look at the influence of one of the three digital channel selection filters on the subcarrier values; the so called *initial*

design. table 3.2 show the parameters that where used for the channel selection filter. See [33] for more information about the design of the filter. figure 3.15 shows the transfer function of the filter. The transfer function shows clearly a wobblyness in the frequency band of interest; it will certainly disturb the subcarrier values. Luckily the disturbance is constant and known, so it is easy for the receiver to compensate for this distortion.

Table 3.2: Digital channel selection design parameters (see [33])

Sample frequency	80 MHz
Pass-band frequency	8.28125 MHz
Stop-band frequency	11.71875 MHz

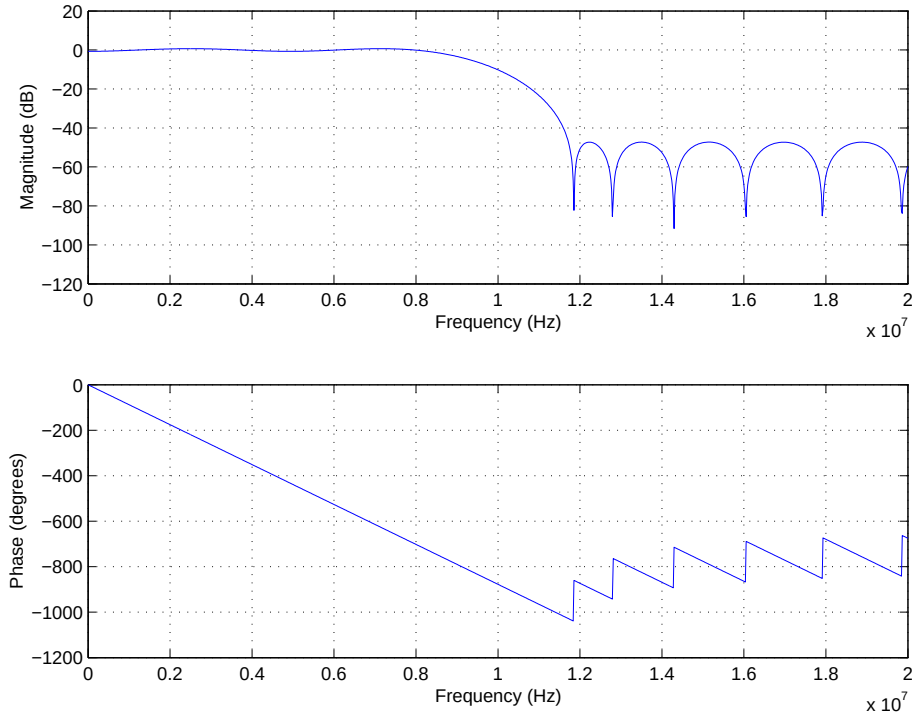


Figure 3.15: Transfer function of the *initial design* digital channel selection filter (see [33])

After the filtering of the signal with the digital channel selection filter the signal is *down sampled* (see [13]) to $f_{sample_R} = 20$ MHz.

3.6 Digital hardware architecture of the SDR receiver

When designing an algorithm, we often start with an ideal mathematical model of the functionality that the algorithm should perform. At this stage, ideal mathematical operations with *infinite* precision are assumed. However, an implementation of a HiperLAN/2 receiver for the software defined radio project will use *arithmetic logic units* (*ALU's*) to perform operations like additions, multiplications and others. These ALU's will be implemented in hardware and will not work with infinite number precision, because the numbers are represented with a *finite* number of bits.

The outcome of operations applied to binary numbers, can not always be represented by the finite number of bits; the result is *rounded* or *truncated*. This will cause an addition of noise to each operation, that an algorithm describes; the outcome of the algorithm will be different than the intended outcome.[13] This error (or propagation of the error) in the outcome of an operation is called *computational noise* (see [18]) and will distort signals in the receiver.

It is not easy to determine the influence of computational noise on the bit error rate of the HiperLAN/2 system, because not only the subcarrier values are disturbed by computational noise, but also synchronization algorithms etc.

In this section we will discuss the modelling of computational noise. Since the receiver model will also be implemented in software –and thus will be using an ALU, to calculate the outcome of operations–, the computational noise must be simulated using the simulator's ALU. This limits the maximum precision, that can be simulated.[14]

In the next section a general binary number representation will be explained. This representation can be used to represent relevant *number types* for the HiperLAN/2 receiver implementation. section 3.6.5 describes the model of the ALU used for simulation purposes to determine the influence of computational noise.

3.6.1 General binary number representation

Digital number representations that are used by ALU's, to perform operations, are groups of bits, in which each bit has a distinguished value it represents. There are several methods how these bits can be ordered in memory. In modelling the HiperLAN/2 receiver we are only interested in *truncation and rounding errors* and not in the *exact implementation* of numbers and their ALU's. Therefore we will use a general binary number representation.

In general, binary numbers can be represented by a bit group

$$s_0\, m_0 m_1 \ldots m_i \ldots m_{n-1}\, s_1\, e_0 e_1 \ldots e_j \ldots e_{p-1} \tag{3.44}$$

with:

[13] In example we want to calculate: $y = (a+b)\,c$, but the result we get is $\hat{y} = (a+b+\eta_1)c+\eta_2$.
[14] A higher *precision* means that the error caused by computational noise is smaller.

$m_{0\ldots n-1}$ – n mantissa bits

$e_{0\ldots p-1}$ – p exponent bits

$s_{0,1}$ – signs bits of the mantissa and exponent

The group of bits from equation 3.44 evaluates to the *decimal* value:

$$\pm(m_0 m_1 \ldots m_g \,.\, m_{g+1} \ldots m_{n-1})_2 \; 2^{\pm(e_0 e_1 \ldots e_{p-1})_2} \tag{3.45}$$

Usually, a point is assumed at a certain position in the mantissa bits. In this document I define the position of the point g to be the number of the mantissa bit that is directly in front of the point. Because there are no bits reserved in the number itself for indicating the position of the point, the point position is always implicitly defined for a certain ALU. The operation $(.)_2$ is defined as (see [19]):

$$(b_0 b_1 \ldots b_g \,.\, b_{g+1} \ldots b_{v-1})_2 = 2^{-v+g+1} \sum_{i=0}^{v-1} b_i \; 2^{v-i-1} \tag{3.46}$$

In example $(10.01)_2$ evaluates to ($v = 4$ and $g = 1$):

$$(10.01)_2 = 2^{-2} \; (8 + 1) = 2.25 \tag{3.47}$$

The general description of binary numbers given in equation 3.44, can be used to represent number types – relevant to HiperLAN/2 physical layer model–, known from the programming world (see [19]):

- Integer numbers
- Fixed point numbers
- Floating point numbers

These types will be discussed in the next sections.

3.6.2 Integer numbers

Integers represent a limited selection of signed natural numbers (for example numbers like -43 and 56). This type of numbers can be represented by the general notation in equation 3.44 by setting the number of exponent bits to zero ($p = 0$). The second sign bit s_1 is not used and the location of the point will be set to the most right position: $g = n - 1$.

The maximum value that can be represented, is:

$$2^n - 1 \tag{3.48}$$

and the minimum value:

$$-2^n + 1 \tag{3.49}$$

Note that the value 0 can be represented twice (with positive sign and negative sign).

3.6.3 Fixed point numbers

Fixed point numbers differ from integers by the location of the point. The point is placed somewhere within the n mantissa bits instead of at the most right position. This means that fixed point numbers can represent a limited selection of real numbers. The numerical distance between two neighboring fixed point numbers is always equal, namely 2^{n-g-1}.

3.6.4 Floating point numbers

This number type is used to represent a limited selection of real numbers. This number type is build of a fixed point number and an integer representing an exponent value. Usually the point is located before the first mantissa bit ($g = -1$). Although other mantissa and exponent combinations are valid, the most efficient way of choosing an exponent number is that the mantissa bit m_0 is *always* **1**. In that case, the most bits in the mantissa are left to represent a number. If the bit m_0 is always **1**, it is a waste of space to store this bit in memory. This bit can better be used to gain extra precision on the number. When this is done the first **1** in the number is called *hidden bit* and the number is assumed to represent the decimal value (see [19]):

$$\pm(.\mathbf{1}m_0m_1 \dots m_n)_2 \; 2^{\pm(e_0e_1 \dots e_p.)_2} \tag{3.50}$$

A drawback of the hidden bit is that the decimal number 0 can not be represented. In this report we will assume that one of the two combinations for 0 in the integer exponent number is used to represent the number 0.

The numerical distance between two neighboring floating point numbers is not always equal. This distance depends on the decimal value of the exponent.

The maximum number that can be represented, is calculated by setting all n mantissa bits and all p exponent bits to **1**:

$$2^{-m}(2^{\sum_{j=0}^{p-1} 2^{p-j}})(\sum_{i=0}^{n-1} 2^{n-i}) \tag{3.51}$$

This equation can be simplified to:

$$2^{-m}2^{D}(2^m - 1) \tag{3.52}$$

With D the upper limit of the exponent, defined as (see equation 3.48)

$$D = 2^p - 1 \tag{3.53}$$

Note that integers and fixed point numbers are special cases of floating point numbers. This will be used in the next section to make a model of truncation and rounding that takes place in an ALU.

3.6.5 Arithmetic logic unit model

In the sections above we have seen a general binary number description, that can be used to describe integers, fixed point an floating point numbers. In this section I will describe a general algorithm that can simulate an ALU on a ALU with a *higher* precision.

First of all we must prove that all possible numbers on ALU A, can be represented on ALU B, that has a higher precision. This proof is quite trivial and can easily be seen from equation 3.45:

$$C_A[x] = \pm(.m_0 m_1 \dots m_{n_A-1})_2 2^{\pm(e_0 e_1 \dots e_{p_A-1})_2} \tag{3.54}$$

$$C_B[x] = \pm(.m_0 m_1 \dots m_{n_B-1})_2 2^{\pm(e_0 e_1 \dots e_{p_B-1})_2} \tag{3.55}$$

Where:

$C_A[x]$ – representation of x on ALU A

$C_B[x]$ – representation of x on ALU B

$C_A[x]$ can be mapped on $C_B[x]$, if $n_B \geq n_A$ and $p_B \geq p_A$, namely

$$C_A[x] = \pm(.m_0 m_1 \dots m_{n_A-1}\mathbf{00}\dots)_2 2^{\pm(\mathbf{00}\dots e_0 e_1 \dots e_{p_A-1})} = C_B[x] \tag{3.56}$$

The last $n_B - n_A$ mantissa bits and the first $p_B - p_A$ exponent bits of $C_B[x]$ are set to zero.

To simulate truncation and rounding errors, we must truncate or round a number with a higher precision $C_B[x]$ to $C_A[x]$. Note that we use the *decimal* representation of $C_A[x]$ and $C_B[x]$ in the remainder of this section.

The following algorithm is capable of mapping $C_B[x]$ on $C_A[x]$:

- Determine the sign's of the mantissa and the exponent (s_0 and s_1 in equation 3.44)
- Determine the exponent of $C_B[x]$:

  $$(exponent)_{10} = ceil(log_2(|C_B[x]|)) - g \tag{3.57}$$

  The resulting exponent should fall in the range $-D \leq exponent \leq D$ (see equation 3.53), otherwise an underflow value *0* or an overflow value of equation 3.51 should be returned.
- Calculate the number that has to be represented by the mantissa, using the following formula:

  $$(mantissa_B)_{10} = \frac{|C_B[x]|}{2^{exponent_B}} \tag{3.58}$$

- Calculate the closest mantissa that can be represented by $C_A[x]$ to $mantissa_B$ of equation 3.58. $Remainder_B$ will be used to keep track of the difference between $mantissa_A$ and the number to represent. At the start it will be set to:

$$remainder_B = mantissa_B \tag{3.59}$$

Repeat for all n_A bits:

$$2^{n_A - i} \geq remainder_B \tag{3.60}$$

with i, representing the current bit number. If this is valid, then calculate the new remainder of the number $C_B[x]$:

$$remainder_B = remainder_B - 2^{n_A - i} \tag{3.61}$$

and update the mantissa of $C_A[x]$:

$$mantissa_A = mantissa_A + 2^{n_A - i} \tag{3.62}$$

- When a rounding machine is used, the value of $mantissa_B$ should be compared with the decimal value of one bit to the right of the last bit:

$$mantissa_B \geq 2^{-n_A - 1} \tag{3.63}$$

If this is valid, $mantissa_A$ should be updated:

$$mantissa_A = mantissa_A + 2^{-n_A} \tag{3.64}$$

- Finally calculate the truncated or rounded number $C_A[x]$ of $C_B[x]$, after restoring the mantissa sign:

$$C_A = \pm mantissa_A 2^{exponent_A} \tag{3.65}$$

The above described algorithm is implemented in C++ and is used to simulate ALU's on a simulating machine with a better machine precision. After every operation and after loading a number into memory the algorithm above is executed. The source code is printed in [7] section A.3.1 and A.3.2. The model also keeps track of the number of operations that has been carried out.

3.7 Conclusion

This chapter discussed signal distortions to the transmitted HiperLAN/2 signal caused by the indoor radio channel, the receiver hardware and the digital channel selection filter.

The following conclusions are drawn in this chapter:

- Reflection, scattering and diffraction cause the receiver to receive attenuated and delayed versions of the transmitted signal. This is modelled with the Rayleigh channel model. Literature states that this is a suitable model of the HiperLAN/2 indoor radio channel (see [22], [25] and [26]).

- The maximum delay spread in the HiperLAN/2 transmission band is typically 130 ns. The cyclic prefix of a OFDM symbol is long enough to combat inter-symbol interference caused by the delay spread.
- Since the HiperLAN/2 terminals or objects in the proximity of the terminals will only move at walking speed, the Doppler shift ($\approx$ 42 Hz) will be negligible to the allowed carrier frequency offset of 250 KHz. The movements will causes the channel to change over time. A channel equalizer in a receiver implementation should be adaptive and should be updated at least once in 10 ms. The channel can be described at baseband level, thus the channel equalizer may also operate at baseband level.
- The receiver hardware creates baseband samples of the received signal.
- Impurities in the down mix frequency can be divided in: frequency offset, common phase offset and phase noise. A frequency offset causes local inter-subcarrier interference locally, while phase noise causes inter-subcarrier interference with all subcarriers. Common phase offset causes a phase rotation equal to all subcarriers. The receiver should compensate for the distortions made.
- The analog to digital converter introduces quantization noise. The anti-aliasing filter distorts the subcarrier values.
- A difference in sample length between HiperLAN/2 transmitter and receiver causes symbol window drift. This cause a phase rotation of the subcarrier values that is dependent on the subcarrier number.
- The digital channel selection filter gives rise to errors in the subcarrier values. Since these disturbances are known, the receiver can easily correct the errors.
- The receiver uses an ALU to perform calculations. The ALU has a finite precision. This causes computational noise to signals in the receiver. A model of an ALU has been presented, that can simulate computational noise.

Chapter 4

Receiver Model Algorithms and Implementation

4.1 Introduction

In the previous chapters is discussed how bits are converted to a signal at a specified carrier frequency and how this signal can be distorted, by the radio channel and the hardware of transmitter and receiver. This chapter describes the "cures" of these distortions and the implementation of these solutions in the receiver model. In chapter 5 the implementations will be put to the test.

In section 4.2 the architecture of the demodulation part in the receiver is outlined (see also figure 3.1). It summarizes what functions should be implemented, based upon the finding of chapters 2 and 3 and how the execution order of those functions is decided.

The remaining sections in this chapter explain the functions that are implemented in the receiver model.

4.2 Receiver architecture

In chapter 2 the functionality of the physical layer on the transmitter side is discussed. The receiver not only has to convert the received signal to data bits by performing the inverse of the transmitter, but it also has to try to inverse distortions caused by radio channel and hardware of transmitter and receiver. In this section we will deduce the execution order of the not yet described operations in the receiver.

The receiver can roughly be divided into two parts:

- *Time domain part.* In the first stage of the transmitter, signal functions

will be time domain functions.

- *Frequency domain part.* In the second stage of the receiver signal functions will be frequency domain functions, because of the IDFT that is necessary to retrieve complex subcarrier values.

Most operations can be performed in time domain *and* in frequency domain. Consider as an example *frequency offset correction*. In the time domain, a frequency offset can be corrected by multiplying the received signal with a complex time-dependent power of e, while in the frequency domain a frequency offset can be corrected simply by shifting the spectrum (see section 3.3.1). In the latter case a frequency offset can only be corrected in steps equal to the frequency resolution of one IDFT bin. In practice we might want to be able to correct in smaller steps.

The location of the functions in the receiver architecture in this study will be based upon a trade-off between the necessary *resolution* that must be reached for a certain correction and the solution with the *minimum* number of operations. By deciding the *execution order* of the functions, we will also try to keep corrections *independent* of each other. Note that to a large extent the order of the functions is already dictated by the definition of the transmitter in chapter 2.

Figure 4.1 shows the receiver architecture that is proposed in this study. The following sections discuss the reason for the location of the functions as well as the algorithms behind the functions.

From chapters 2 and 3 can be concluded that an implementation of the physical layer in the HiperLAN/2 receiver should at least contain the following functions:

- Synchronization function
- Frequency offset corrector
- Phase offset corrector
- Channel equalizer (partly implemented in the model)
- Inverse OFDM
- Demapping
- Deinterleaving (not implemented in the model)
- Viterbi-decoder (not implemented in the model)
- Descrambling (not implemented in the model)

In figure 4.1 other functions are also shown. Those will explained below.

After the functions were implemented, the receiver model was tested. It passed its functional test (see [28]).

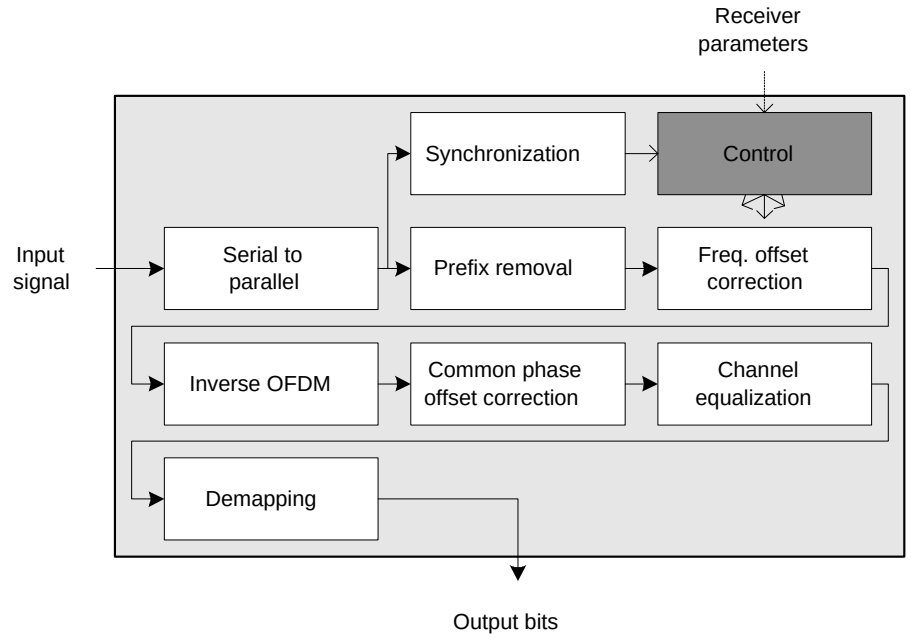


Figure 4.1: Demodulator part of the receiver (see figure 3.1)

4.3 Serial to parallel conversion

One OFDM symbol is –in the ideal case $f_{sample} = 20$ MHz– represented by 80 input samples, so once per 80 samples the receiver will have enough information to demodulate the received samples and output the resulting bits. The main demodulation function of the receiver, an IDFT, works with *parallel* data instead of the serial data generated by the hardware of the receiver. Hence we will create a vector of at least 80 received samples, containing the oldest sample at the top of the vector and the newest sample at the bottom.

At this point in the receiver, the timing of the OFDM symbols is unknown and therefore each incoming sample has to be stored by throwing out the oldest sample in the vector, shifting all samples one position upward and storing the received sample at the bottom of the vector. A *cyclic* buffer can perform this function efficiently, because the index of the samples change rather than shifting the samples in memory. This is illustrated in figure 4.2. A cyclic buffer needs one extra storage location compared to a vector to store the location of the top of the vector.

The source code of the serial to parallel conversion using a cyclic buffer is printed in [7] section A.3.1.

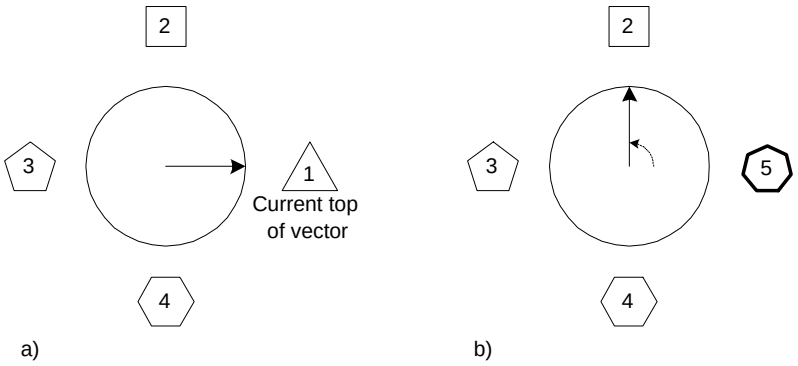


Figure 4.2: Illustration of a cyclic buffer; a) Initial state. b) A new sample overwrites the oldest sample and the top of the vector is moved one position

4.4 Synchronization

Synchronization between transmitter and receiver is an important issue for HiperLAN/2. A small timing error can cause large errors in the demodulated bits (see [30]). A HiperLAN/2 transmission burst is always proceeded by a known sequence of special OFDM symbols –the preamble– as was discussed in section 2.8. This information can be used to detect the beginning of an OFDM symbol train, that contains information for higher protocol layers.

The task of the synchronization entity can be divided into three separated functions: detecting the *beginning of a burst*, detecting the *preamble parts* and *tracking the symbol window drift*. The synchronization entity signals the controller of the receiver when its state should be changed. We define the following receiver states:

- START: the initial state of the controller in the receiver. It is waiting for a transmission to begin.
- PREAMBLE A: A possible beginning of a transmission has been detected. The receiver is trying to find preamble section A in the incoming samples.
- PREAMBLE B: The receiver detected preamble section A in its input samples. It is trying to detect preamble section B in the incoming input samples. There are two types of preamble section B (see section 2.8), so this receiver state will be divided in two separated states: PREAMBLE B LONG and PREAMBLE B SHORT.
- PREAMBLE C: The receiver detected preamble section B in its input samples. It is trying to detect preamble section C.
- DATA: Preamble section C was detected. The receiver is currently receiving regular OFDM symbols that need to be demodulated.

The state transitions differ per burst type (see section 2.8). Figure 4.3 shows the state transitions that take place in the controller depending on synchronization information.

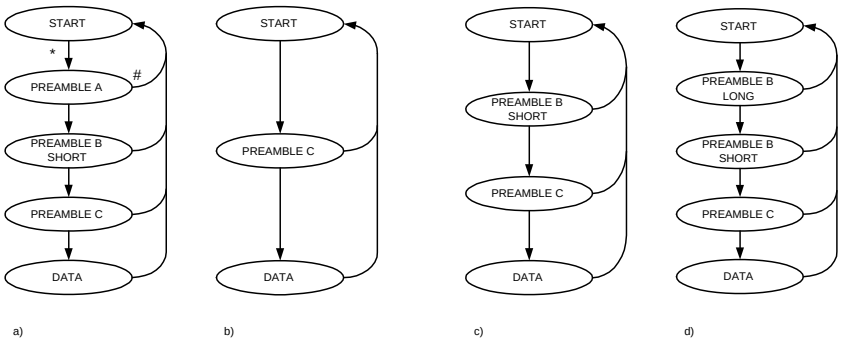


Figure 4.3: Receiver state transition for: a) Broadcast burst, b) downlink burst, c) uplink burst with short preamble and d) uplink burst with long preamble or direct link burst. When a transmission or preamble has been detected (*), the state advances. When a timeout occurs (#) the state is reset to state START

The next sections will discuss how the beginning of a burst can be detected, how the preambles can be detected and finally how the synchronization function can keep track of symbol window drift.

4.4.1 Detecting a transmission

The beginning of a transmission can be detected by measuring the input signal power. When this power is higher than a predefined noise power threshold, a burst might be transmitted. The state of the transmitter is advanced to an appropriate state (see figure 4.3).

To prevent that the receiver triggers on possible high power noise spikes, the average power over a few samples should be compared to a predefined threshold. In the receiver implementation the average power of the last 16 input samples is used.

4.4.2 Detecting preamble sections

After a possible start of a transmission has been detected, the preambles must be detected. At this point in the receiver no distortions are yet corrected, so a robust method must be chosen to detect the preambles. In literature the detection of preambles is often done by a *matched filter*.

Preamble sections A and B contain parts of 16 samples that are equal to each other, apart from sign changes that are not considered, and therefore a 16 sample wide matched filter should be used (see figure 4.4).

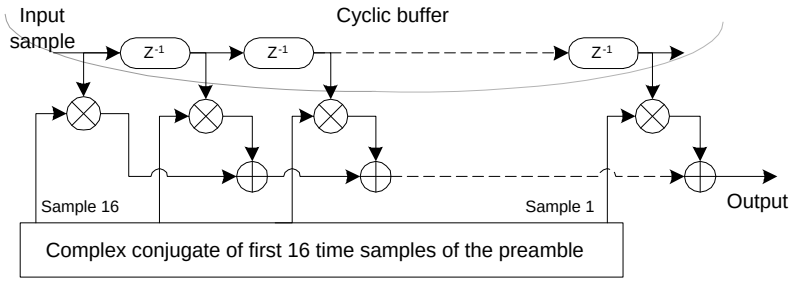


Figure 4.4: Matched filter for detection of 16 sample parts in preamble sections A and B (see section 2.8). The normalization step, discussed in the text, is not drawn

Before the outcome of the matched filter is calculated, both input samples and preamble samples calculated in the receiver are normalized. This prevents that the outcome of the filter is different, when the power of the input signal changes. Mathematically we calculate the absolute value of the *cosinus* of the angle between a vector $\vec{v}_{input}$, that contains the last 16 input samples and a vector $\vec{v}_{preamble}$, that contains the first 16 samples of preamble section A or B. The *cosine rule* is given by ($\vec{a}$, $\vec{b} \in \mathbb{C}^n$):

$$\cos(\vartheta) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}|\ |\vec{b}|} \tag{4.1}$$

With (b_i^* is the complex conjugate of b_i):

$$\vec{a} \cdot \vec{b} = \sum_{i=0}^{n-1} a_i\, b_i^* \tag{4.2}$$

and

$$|\vec{a}| = \sqrt{\vec{a} \cdot \vec{a}} \tag{4.3}$$

Thus the matched filter calculates:

$$\left| \frac{\vec{v}_{input} \cdot \vec{v}_{preamble}}{|\vec{v}_{input}|\ |\vec{v}_{preamble}|} \right| = |\cos(\vartheta_{v_{input}, v_{preamble}})| \tag{4.4}$$

The more correlation exists between the received samples and the expected samples, the closer the angle $\vartheta_{v_{input}, v_{preamble}}$ will approach zero, and the closer the output value of the filter will be to 1.0.

The output of the filter, when it is used to find preamble section A in the input samples, is drawn in figure 4.5. The figure shows clearly the spikes to the value 1.0, when the preamble section parts have been matched. When the output of the filter is compared with a *threshold* value less than 1.0 for detecting a match, the system can be made more robust against distortions in the input signal. Figure 4.6 shows the output of the filter for a signal to noise ratio[1] of 1.5 dB; we clearly must be satisfied with less correlation between input samples and expected input samples, but still the preamble section parts can be detected.

[1]Signal to noise ratio is defined as $\sigma^2_{preamble}/\sigma^2_{noise}$.

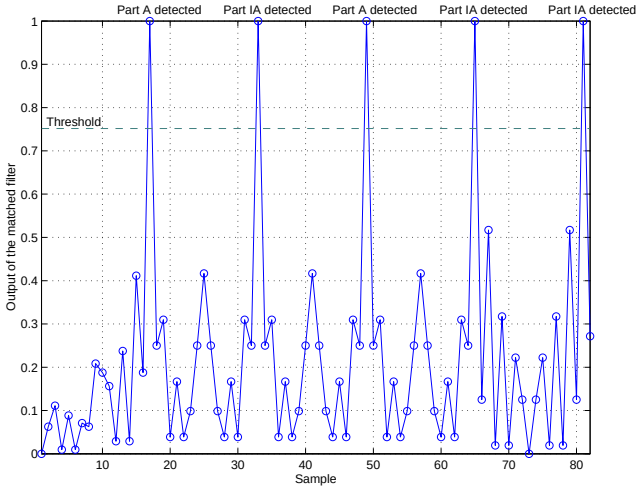


Figure 4.5: Absolute value of the output of matched filter, when it is used to detect preamble A section parts A IA A IA IA. The filter output is compared with a threshold value to make a decision. In this figure any threshold between ≈ 0.6 and 1 will correctly detect the preamble sections

When the distance between the filter output peaks is larger or smaller than expected[2], the receiver state is reset to state START.

Preamble section B can be detected in the same manner as preamble section A. Preamble section C can be detected by using a 32 sample wide matched filter and detecting three times the cyclic prefix part of C. The source code, used in the receiver model for detecting the preamble sections, is printed in [7] section A.5.3.

Once an entire preamble section is received, another function in the receiver can use the information in the preamble section to gather information about distortions it should correct.

4.4.3 Tracking symbol window drift

section 3.4.1 discussed the cause of symbol window drift. The cyclic prefix of a data OFDM symbol contains a copy of the last 16 samples of the useful data part of the symbol. This extra information can be used to find the beginning of the symbol in a vector of input samples. In the model described in this chapter, I use the same principle as for detecting the preamble section parts (see section 4.4.2): a matched filter. This time the transmitted data is not known to the receiver, therefore the correlation is calculated between 16 samples and 16 samples received 64 samples earlier.

It would be a waste of calculation power to calculate this correlation for

[2]The implemented model allows one sample time deviation of the expected distance, since we expect a drift of 1 sample per 300 OFDM symbols (see section 3.4.1). Note that the sample frequency of the receiver hardware can not be adjusted by the software.

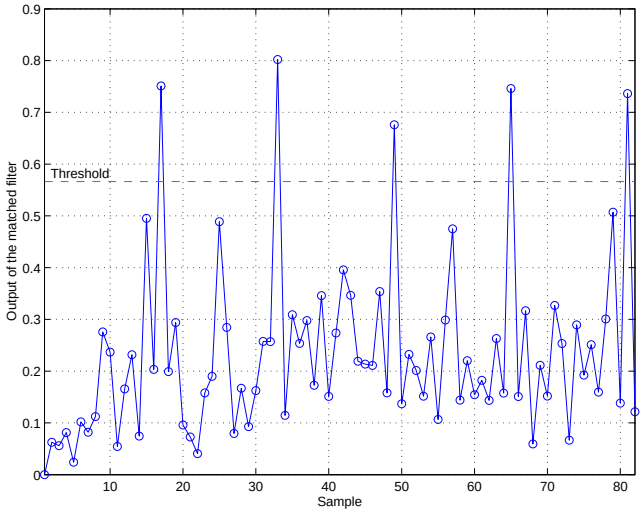


Figure 4.6: Absolute output of the matched filter, while detecting preamble section A ($SNR = 1.5$ dB). Less correlation exists between input samples and expected input samples

each incoming sample and therefore an *estimate* is made for the location of the symbol start: the **Expected start of symbol**. The controller uses this estimate for the moment in time to trigger a symbol window tracking and demodulation cycle.

So far the *length* of the cyclic buffer –described in section 4.3– has not been discussed. In this report I have chosen for a length 96 cyclic buffer. This gives the symbol window the opportunity to move eight samples in both directions, since a symbol has a length of 80 samples.

The symbol window tracker provides the following information to the controller of the receiver:

Start of symbol – Index in the cyclic buffer that contains the first sample of the prefix of the current symbol. This index is found using a matched filter, which will be described below.

Decode stop – Tells the receiver when the next demodulation cycle is due. This also provides an estimate for the start of the next symbol (see below).

Difference – This represents the difference between estimated start and *found* start of the symbol. Large differences tell the controller that synchronization failed.

Figure 4.7 outlines the contents of the cyclic buffer when the demodulation cycle is due. It also shows the (fixed) relation between the decode stop, the expected start of the symbol and the search area for the start of the symbol. Note that in this approach there is always a delay of about eight samples after the last received sample of a symbol and the actual demodulation.

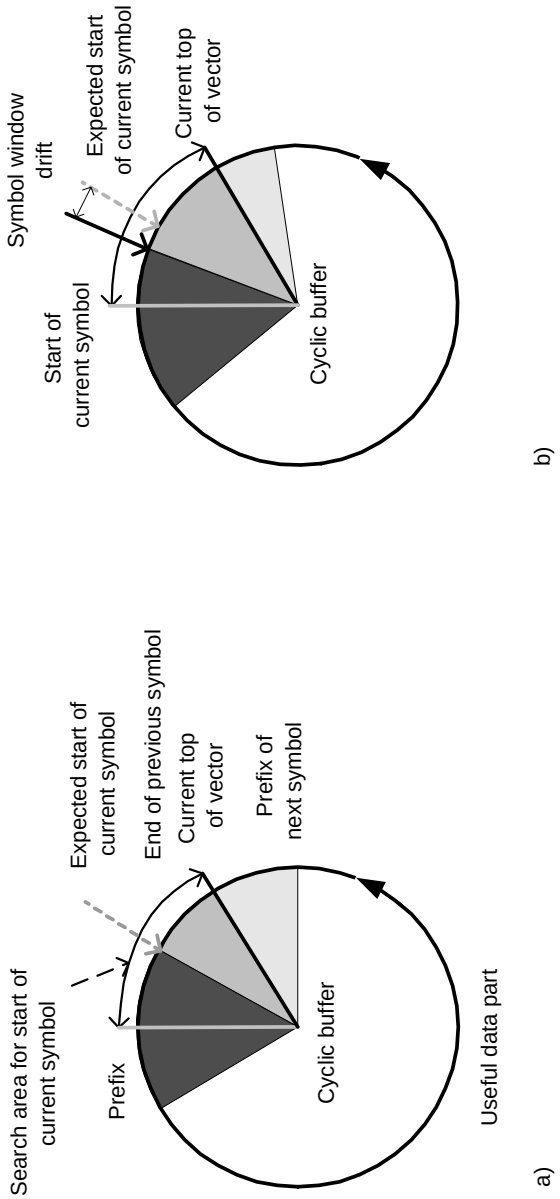


Figure 4.7: Contents of cyclic buffer at the moment of searching for the symbol window. In a) no symbol window drift is present. The expected start of the symbol is equal to the start of the symbol. b) shows the contents of the buffer if symbol window drift is present

Sixteen times the outcome of the matched filter is determined, each time advancing one sample[3]. The maximum correlation indicates the location of the prefix: the **Temporary start of the symbol**. Noise and other distortions may cause that the correlation at locations close to the start of the symbol is higher than at the start of the symbol[4]. This results in an incorrect temporary start of the symbol. The number of samples at which the symbol window might move, is about one sample per 300 OFDM symbols (see section 3.4.1). In the receiver model discussed in this report, the movement of the symbol window is restricted by calculating the mean displacement in the last 30 OFDM symbols.

The symbol window tracker sets the start of the symbol equal to the expected start of the symbol, corrected by the mean displacement and the next decode stop to eight samples clockwise of the start of the symbol.

The source code of the symbol window tracker is printed and described in [7] sections A.5.3 and A.5.4. This function can be enabled or disabled in the receiver model, to investigate its functioning.

4.5 Prefix removal

At this point in the receiver, the location of the first sample of the data symbol is known, namely **Start of symbol** as discussed in section 4.4. At a distance of $T_{CP} = 16$ samples the useful data part of the symbol begins. The next $T_U = 64$ samples are copied to a length 64 cyclic buffer. All functions after this prefix removal function will operate on the new cyclic buffer, since it is a waste of calculation power to correct errors in the prefix. The cyclic prefix of the symbol will only be used to keep track of frequency offset (see next section) and could possibly be used in channel equalization.[5]

4.6 Frequency offset estimation

In section 3.3.1 we saw that a difference in mix frequencies between transmitter and receiver causes inter-subcarrier interference. We assumed that the frequency offset does not change over time. In this section a method will be presented that enables the receiver to compensate (partly) for frequency offsets, using information that can be retrieved from the received preamble sections. The frequency offset correction function has two functions:

- Determination of the frequency offset Δ'_f
- Correction of the frequency offset

In the following sections these tasks will be discussed.

[3] In figure 4.7 this is denoted as "search area for start of current symbol".
[4] Note that the prefix and the end of the symbol are both different distorted.
[5] This function is not implemented in the channel equalizer in this report.

4.6.1 Measuring frequency offset

The preamble sections are –just like regular OFDM symbols– distorted by a frequency offset. Since the sections contain known subcarrier values, the data can be used to measure effects of a frequency offset. We will denote the measured frequency offset by Δ'_f. This measurement should be as close as possible to Δ_f.

There are two approaches to this problem; the frequency offset can be determined in the frequency domain and in the time domain. First the frequency domain method will be discussed, followed by the time domain method that has been implemented in the receiver model.

In the frequency domain there is a direct relationship between measured power in a subcarrier caused by inter-subcarrier interference and the frequency offset present (see figure 3.10 and equation 3.27).[6] Preamble sections A and B (see section 2.8) are suitable for this kind of frequency offset detection, since they contain loaded subcarriers surrounded by not-loaded subcarriers (see [24]).

An estimation of the frequency offset can be made by measuring the average power[7] in the subcarriers next to the loaded subcarriers in preamble sections A and B, and use the result of figure 3.10 to determine the frequency offset Δ'_f. This method is *not* implemented in the model of the physical layer of the receiver, because we have seen in chapter 3 that inter-subcarrier interference also can be caused by –in example– phase noise.

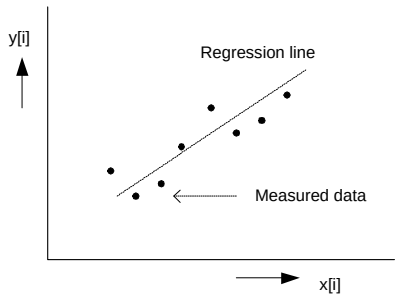


Figure 4.8: Regression line in measured data. The slope of the line is calculated with equation 4.6. This is used to determine the average rotation per sample caused by a frequency offset

Another to method to measure frequency offset takes place in the time domain. This method measures the average rotation per sample in preamble sections A and B. Consider the following definition ($a, b \in \mathbb{C}$ and $a, b \neq \mathbf{0}$):

$$\Theta(a,b) \triangleq \arctan\left(\frac{\Re(a)}{\Im(a)}\right) - \arctan\left(\frac{\Re(b)}{\Im(b)}\right) \tag{4.5}$$

This function calculates the *angle* between a and b. We will determine the rotation between the *expected* sample value in a preamble and the *received* input

[6] We assume that other distortions that can cause inter-subcarrier interference, are not present.

[7] To minimize the influence of noise.

sample. Subsequently we will calculate the slope of the *regression line* of the rotation. The regression slope is defined as (see [34] and figure 4.8):

$$\frac{N \cdot \sum_{i=0}^{N-1} x[i]\, y[i] \;-\; \sum_{i=0}^{N-1} x[i] \cdot \sum_{i=0}^{N-1} y[i]}{N \cdot \sum_{i=0}^{N-1} x^2[i] - \left(\sum_{i=0}^{N-1} x[i]\right)^2} \tag{4.6}$$

In which $y[i]$ is the measured rotation $\Theta(\vec{v}_{preamble}[i]\,,\, \vec{v}_{input}[i])$, $x[i]$ is the sample index (thus $x[i] = i$) and N is the number of samples that we use. The equation can be simplified with respect to $x[i]$ and N, namely:

$$\frac{64 \cdot \sum_{i=0}^{63} i\, \Theta(\vec{v}_{preamble}[i]\,,\, \vec{v}_{input}[i]) \;-\; 2016 \cdot \sum_{i=0}^{63} \Theta(\vec{v}_{preamble}[i]\,,\, \vec{v}_{input}[i])}{1397760} \tag{4.7}$$

In this equation we used $N = 64$ for two reasons: the preambles are also subject to the delay spread of the channel, hence we not use the first 16 samples (just like the cyclic prefix to regular OFDM symbols), because they can contain *inter-preamble interference*. And the second reason is a more practical one: the preambles are generated using a 64 sample IFFT and hence it is practical to work with 64 samples.

The result of equation 4.7 is the rotation per sample. This value can easily be used to calculate Δ'_f. Note that other distortions can have influence on the rotation we calculate.

In the receiver model currently only preamble section A is used to get an estimation of the frequency offset. Note that the last 16 samples of this preamble section are sign inverted compared to the previous 16. The rotation for that part should thus be corrected with π.

The model should be extended with the frequency estimation using preamble section B. A solution must be conceived on how to determine the frequency offset when *only* preamble section C is transmitted. Probably, the frequency domain method discussed above can be applied to subcarriers -27 and 27, because those are not used to carry a value.

4.6.2 Correcting frequency offset

In the introduction of this chapter we have already seen that the correction for a frequency offset should take place in the time domain, because in that case it is possible to correct for smaller frequency deviations than one of subcarrier spacing Δ_f.

The frequency offset is corrected by multiplying all input samples with:

$$e^{-\jmath 2\pi f'_\Delta t} \tag{4.8}$$

before further processing.

The source code of the frequency offset corrector implemented in the physical layer model of the receiver, is printed in [7] section A.5.5. This function can be enabled or disabled in the receiver model, to investigate its functioning.

Note that this implementation does *not* correct for a phase offset that is common to all input samples, although this is necessary for the correct operation of the frequency offset corrector. The common phase offset should be removed by the common phase offset corrector (see section 4.8).

4.7 Inverse orthogonal frequency division multiplexing

At this point in the receiver, 64 time domain samples are extracted that (most likely) represent the useful data part of the OFDM symbol that has to be demodulated. But before the demodulation can take place, the subcarrier values must be retrieved from the useful data part. This can be done by applying a *fast Fourier transform* (*FFT*) to the vector containing the 64 samples (see also 4.5). The FFT is the *inverse* operation of the IFFT that already has been discussed in section 2.7.2. The FFT efficiently implements a DFT, namely

$$\hat{f}_n[x] = \sum_{m=0}^{N-1} \widetilde{s_n}[m] e^{-\jmath 2\pi \frac{xm}{N}} = \frac{1}{N} \cdot DFT(\widetilde{s}_n[m]) \tag{4.9}$$

With $x = 0, ..., 63$ and

$$\hat{f}_n = [0\, \hat{C}_{n,1}\, \hat{C}_{n,2}\, \ldots\, \hat{C}_{n,26}\, 0\, 0\, 0\, 0\, 0\, 0\, 0\, 0\, 0\, 0\, 0\, \hat{C}_{n,-26}\, \ldots\, \hat{C}_{n,-2}\, \hat{C}_{n,-1}] \tag{4.10}$$

From this vector the 52 subcarrier values can be extracted.

In the receiver model, an *in-place* FFT has been implemented (see [35]). The advantage of this algorithm is that it uses the same vector for input as well as output. This saves memory spaces. The FFT implementation uses *radix two butterflies* to calculate its outcome (see [35]). The source code of this operation is printed [7] section A.5.8.

4.8 Common phase offset correction

Common phase offset occurs when mixers in transmitter and receiver do not have the same phase at a given time.[8] In section 3.3.2 we saw that this phase offset is common to all subcarriers.

[8] In this section we assume that there is no frequency offset present in the system, for example $f_\Delta = 0$ or that the system corrected the frequency offset correctly ($\Delta'_f = \Delta_f$). A phase difference will thus not be caused by a frequency offset.

The common rotation of the subcarriers can be determined by calculating the mean rotation of the pilot carriers (see section 2.7.1) compared to their expected values, using equation 4.5. Keep in mind that a timing error causes a *distributed* rotation of all subcarriers (see equation 3.43).

In figure 4.9 an example is given of the phase offset correction. In the example QPSK is used as subcarrier modulation technique. The uncorrected subcarriers are rotated clockwise. This will introduce large bit errors.

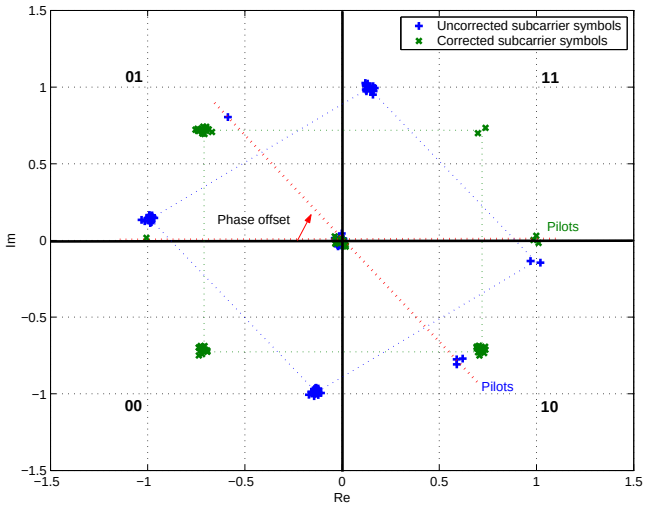


Figure 4.9: Example of the phase offset correction for QPSK

The source code of this function can be found in [7] sectio A.5.6. This function can be enabled or disabled in the receiver model, to investigate its functioning.

4.9 Phase noise correction

In [32] phase noise is discussed. The author states that phase noise can not be corrected in the receiver *at reasonable costs* and hence we will not try to find a solution for this signal distortion in this report.

4.10 Channel equalization

One of the conclusions of chapter 3 was that a HiperLAN/2 receiver should have a channel equalizer. The transfer function of the indoor radio channel can be written at baseband level (see equation 3.18). This means also that the channel can be equalized in the frequency domain, thus after the inverse OFDM (see [30]).

A channel equalizer is partly implemented in the model of the physical layer of the receiver. Since the coherence time of the channel is about 10 ms (see section 3.2.4) and a burst has a duration of 2 ms, I did not implement an adaptive channel equalizer. When the HiperLAN/2 channel models are implemented, future work should decide whether this is a correct decision.

The channel estimator in the receiver model uses preamble section C to determine an estimation of the channel. This preamble is modulated BPSK (see section 2.8). The following equation is used ($i = -26, ..., -1, 1, ..., 26$):

$$\frac{1}{\hat{H}_l} = \vec{V}_{preamble}[l] / \vec{V}_{input}[l] \tag{4.11}$$

With $\vec{V}_{preamble}$ is the subcarrier vector of preamble section C and $\vec{V}_{input}$ is the FFT of 64 input samples recognized as preamble section C.

Before demapping (see section 4.11) each subcarrier value is corrected with:

$$\check{C}_\gamma = \frac{\hat{C}_\gamma}{\hat{H}_\gamma} \tag{4.12}$$

This channel equalization method is sensitive to noise. In [30] the effects of channel estimation errors is discussed. Note that also preamble section C must correctly be identified.

The source code of the channel estimator is printed in [7] section A.5.10. The channel estimation function can be enabled or disabled in the receiver model.

4.11 Demapping

As discussed in section 2.6, there are four subcarrier mapping techniques: BPSK, QPSK, 16QAM and 64QAM. Each of those techniques has a different number of bits per complex subcarrier symbol. In this section the inverse process of mapping will be discussed.

Consider the received the demodulated subcarrier symbol $\hat{C}_\gamma$. This symbol is distorted by the radio channel and receiver hardware, and is adjusted by the demodulator part of the receiver. The most likely symbol that was transmitted, is probably the symbol to which $\hat{C}_\gamma$ is closest to in distance. The *Euclidian distance* is given by ($a, b \in \mathbb{C}$):

$$|a - b| \triangleq \sqrt{(\Re\{a\} - \Re\{b\})^2 + (\Im\{a\} - \Im\{b\})^2} \tag{4.13}$$

We define all possible subcarrier values for a certain mapping scheme $\mathbf{C}$. One possible symbol out of the list of all possible symbols in denoted by $c_\xi \in \mathbf{C}$. To find the nearest subcarrier value to the received subcarrier value, calculate:

$$i = \arg \min_{\xi=[0,..,2^{N_{BPSC}}-1]} |\hat{C}_\gamma - c_\xi| \tag{4.14}$$

The value i is directly related to the bit pattern associated with the most likely received symbol. This is repeated for all 48 subcarriers.

This method of demapping is called *hard decision, coherent* demapping (see [10] and [14]). The source code is printed in [7] section A.5.7. This is the only function in the receiver that is dependent on the subcarrier modulation type.

4.12 Conclusion

The following functions are implemented in the model of the demodulation part of the receiver (see also figure 4.1):

- *Serial to parallel conversion.* This function is implemented in the model, because the main demodulation function of the receiver, an IDFT, works with parallel data instead of the serial data generated by the hardware of the receiver. The input samples of the receiver are stored in a cyclic buffer. This kind of buffer is efficient, because no shifting of samples is necessary, when the oldest sample must be erased from the bottom and a new one stored at the top.
- *Synchronization.* This function takes care of detecting a transmission burst, detecting the preambles in the burst and tracking symbol window drift. This function uses a matched filter to operate on time domain samples. Demodulation cycles will take place about every 80 received samples.
- *Prefix removal.* After synchronization the useful data part is extracted from the input samples.
- *Frequency offset correction.* Frequency offset estimation can take place in time domain as well as in frequency domain. In this report is chosen for the time domain option. This method uses the preambles in the burst, to determine the average rotation per sample, by calculating the slope of the regression line of all rotations, between received and expected samples in the preamble. The estimation is used to rotate all input samples to the receiver appropriately.
- *Inverse OFDM.* Next the subcarrier values are retrieved from the time domain samples by applying a special form of DFT: the efficient FFT algorithm.
- *Channel equalization.* The implemented receiver model uses preamble section C to determine the channels transfer function. This transfer function is obtained and equalized in the frequency domain.
- *Demapping.* A received subcarrier value is quantized into one of all possible subcarrier values of a mapping technique, by determining the smallest Euclidean distance. Next the bit values associated with the symbol are passed to the output of the receiver model. This is repeated for all 48 subcarriers.

Chapter 5

Model Simulation Results

5.1 Introduction

This chapter presents some simulation results of the HiperLAN/2 physical layer model. First the transmitter and receiver will be connected directly together. In this way we can see how, in the ideal case, signals in the system should look. Two methods for looking at complex subcarrier values will be presented.

The following experiment determines the performance of the model on an AWGN channel. The result is compared to the theoretical expected performance, as was discussed in section 2.10. The experiment is executed for 64-bit floating point numbers, 32-bit fixed point numbers and 16-bit fixed point numbers, to investigate the influence of computational noise (see section 3.6).

In the next conducted experiment, the influence of noise on the frequency offset estimator and the common phase offset corrector is investigated. For this experiment an AWGN channel will be used too.

At the end of this chapter a *real world* experiment will be done. The output of the HiperLAN/2 transmitter is fed to a *signal generator*. The output of the signal generator is sampled with an oscilloscope and passed along to the HiperLAN/2 receiver model.

Throughout this chapter the transmitter model will transmit random bits (with equal probability for a **1** or a **0**), the broadcast burst will be used and 64-bit floating point numbers will be used, unless otherwise noted.

5.2 Ideal channel simulation results

In this section we will have a look at the model, when no distortions like frequency offset or phase offset are present. This gives insight in how the signals in the model should look in the ideal case. We will also determine the computa-

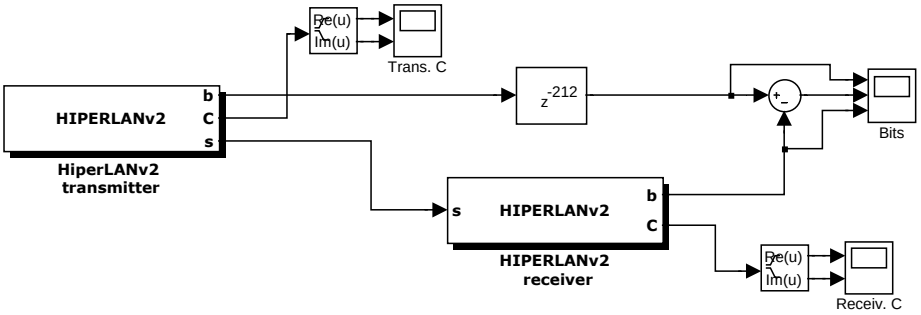


Figure 5.1: Model configuration

tional requirements of the transmitter and the receiver model. The simulation structure that will be used for the experiments in this section, is depicted in figure 5.1.

5.2.1 Visualization of outputs

The experiment set-up has three outputs: the complex subcarrier values that are used to create the OFDM symbols[1] (this is named `Trans. C` in figure 5.1), the complex subcarrier values before they enter the demapping part in the receiver (`Receiv. C`) and the input bits of the transmitter, compared with the demodulated bits (`Bits`).

There are two ways of looking at the subcarrier outputs: the contents of a subcarrier plotted against its *subcarrier number* l or the contents of a subcarrier plotted against *time*. The first method is in fact a frequency spectrum of the received signal. The latter method is called an *eye diagram*.

In figure 5.2 the contents of the subcarriers is plotted against their subcarrier number. The *pilot carriers* (see section 2.7.1) are clearly visible in the figure. QPSK was used as subcarrier modulation method, hence two subcarrier values (besides the pilot values and zero) appear in the plots of $\Re\{C\}$ and $\Im\{C\}$.

In figure 5.3 an eye diagram is drawn of the subcarrier values that are received. In the figure QPSK is used as modulation method. The OFDM symbols that carry data are proceeded by a *broadcast* preamble (see section 2.8) and hence the subcarrier values stay zero for some time.[2] Because preamble section C is used in the receiver model for *channel equalization* (see section 4.10), the subcarrier contents of that preamble is also written to the output `Receiv. C`. This section is clearly visible in the figure.[3] Even the pattern of the pilot values can be distinguished in the figure (see equation 2.13).

[1] In chapter 2 this was denoted as f_n in equation 2.19.

[2] Note that the receiver also has to gather all 80 time samples of the first data OFDM symbol *and* an additional 8 samples to allow symbol window drift (see section 4.4.3).

[3] Preamble section C is BPSK modulated and thus the subcarrier values contain only a *real* component. The configuration of the preamble is discussed in section 2.8.

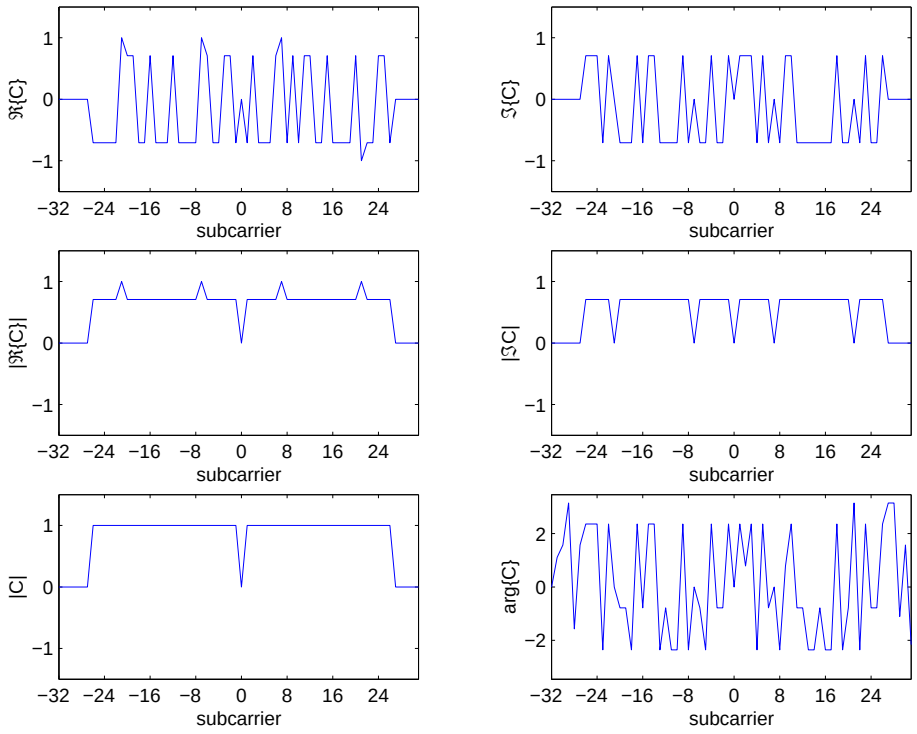


Figure 5.2: Contents of the subcarriers in the receiver (`Receiv. C`) against their subcarrier number. OFDM symbol $n = 0$ is depicted. The pilot carriers are clearly visible at $l = -21, -7, 7$ and 21. QPSK was used as subcarrier modulation method

The two methods of looking at the subcarrier outputs are used throughout this chapter. They form a useful tool to identify the effects of distortions or –when new receiver algorithms are being developed– to spot errors in the receiver model implementation. Another useful visualization tool is to subtract the transmitted bits and the demodulated bits. In this way *bit errors* can be seen quickly. Note that the transmitted bitstream must be delayed to compensate for the delay, that occurs in the receiver (see section 4.4.3). The output `Bits` is shown in figure 5.4.

5.2.2 Computational requirements of transmitter and receiver algorithms

The receiver and transmitter models keep track of the number of numerical operations, like "+" and "/", that is carried out (see section 3.6.5). For this experiment we will use a fixed burst type, namely *broadcast* and the receiver will be set to enable *all* compensations, like frequency offset correction and channel estimation. Next the modulation type of the model will be changed. The transmitter model writes its operation usage statistics to the file `TxInfo.txt` and the receiver to `RxInfo.txt`.

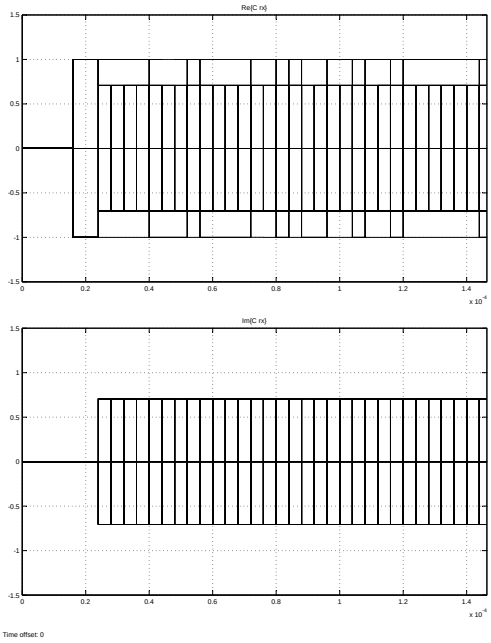


Figure 5.3: Eye diagram of `Receiv. C`

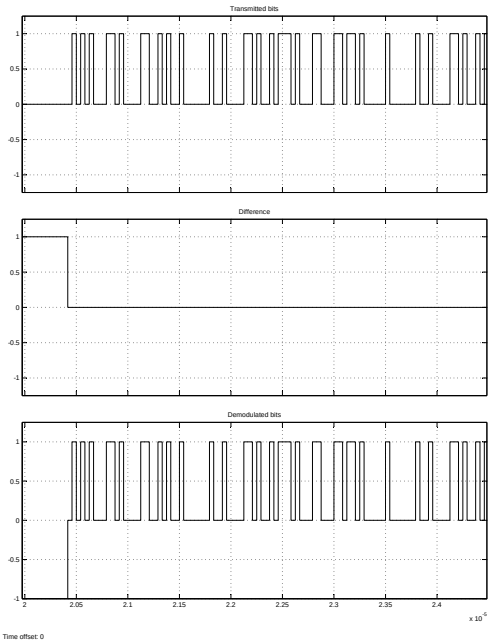


Figure 5.4: Visualization of bit errors. In this simulation no distortions were present and hence no bit errors occur. When bit errors do occur, they will show in the "difference" graph

In table 5.1 the computational requirements *per second* of the transmitter are presented. Note that there are about $250,000$ OFDM symbols in a single second. This means that the IFFT function already preforms $24\,10^6$ complex multiplications and $96\,10^6$ complex subtractions and additions (see [30] and [35]). In table 5.1 we see these results clearly, since for a *complex* addition or subtraction, two *real* additions or subtractions must be calculated. One complex multiplication requires four real number multiplications and two additions. Some of those multiplications disappear to the special operations, since they involve rotations.

Table 5.1: Computational requirements of the transmitter per second. *Special operations* denote square roots, rotations etc. Note that these statistics represent *real number* operations; *complex number* operations are split into real number operations

Modulation type:	BPSK	QPSK	16QAM	64QAM
Additions	$95.616\,10^6$	$95.616\,10^6$	$95.616\,10^6$	$95.616\,10^6$
Subtractions	$128.014\,10^6$	$128.014\,10^6$	$128.014\,10^6$	$128.014\,10^6$
Multiplications	$47.808\,10^6$	$47.808\,10^6$	$47.808\,10^6$	$47.808\,10^6$
Divisions	0	0	0	0
Comparisons	0	0	0	0
Special operations	$191.232\,10^6$	$191.232\,10^6$	$191.232\,10^6$	$191.232\,10^6$

Table 5.2: Computational requirements of the receiver per second (see also comment of table 5.1)

Modulation type:	BPSK	QPSK	16QAM	64QAM
Additions	$587.333\,10^6$	$610.853\,10^6$	$751.973\,10^6$	$1,316.453\,10^6$
Subtractions	$369.392\,10^6$	$416.432\,10^6$	$698.672\,10^6$	$1,827.632\,10^6$
Multiplications	$929.110\,10^6$	$976.150\,10^6$	$1,258.390\,10^6$	$2,387.350\,10^6$
Divisions	$332.916\,10^6$	$332.916\,10^6$	$333.130\,10^6$	$333.130\,10^6$
Comparisons	$197.618\,10^6$	$221.138\,10^6$	$362.258\,10^6$	$926.738\,10^6$
Special operations	$417.248\,10^6$	$417.248\,10^6$	$417.248\,10^6$	$417.248\,10^6$

Table 5.2 shows the computational requirements of the receiver algorithms. The algorithms are clearly dependent on the subcarrier modulation type that is used. In chapter 4.11 we have seen that the *demapping* function is the only function in the receiver model that is dependent on the subcarrier modulation type. Hence can be concluded that this function performs a large part of the calculation that are done. Note that for 64QAM in total $250,000 \cdot 64 \cdot 48 = 768\,10^6$ comparisons per second are made by the demapping function, each involving two multiplications and three additions/substractions. A future study should conceive on a more efficient demapping function.

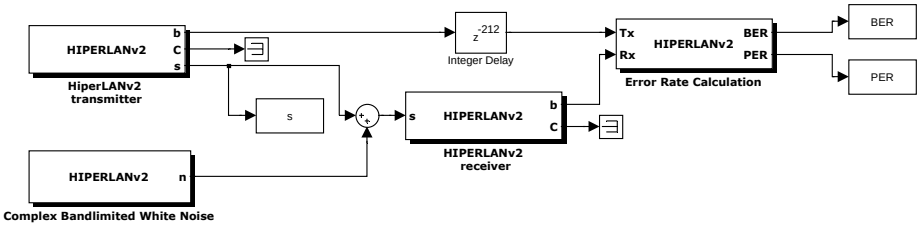
The demonstrator of the SDR project will be using a *general purpose processor* of a personal computer to perform the demodulation tasks in the digital domain (see [2]). Future studies must show whether the proposed processor can meet the computational requirements of the receiver algorithms discussed in this report.

5.3 Comparison theoretical and simulated performance on AWGN channel

Section 2.10 discussed the *expected performance* of the HiperLAN/2 system, when transmitting over an AWGN channel. In this section we will compare the performance of the simulation model with the expected performance. First the AWGN channel experiment will be carried out with the best *number precision*[4] available (64-bit floating point number; see [4] and [3]), followed by a simulation using 32-bit fixed point numbers (as is proposed in [17]) and a simulation using 16-bit fixed point numbers. In all simulations 25,000 OFDM symbols[5] were used to obtain the results.

5.3.1 Experiment configuration

The layout of the model used in the AWGN channel simulations is depicted in figure 5.5. This model layout is stored separately for the four modulation techniques, because this eases the gathering of simulation results.



ment

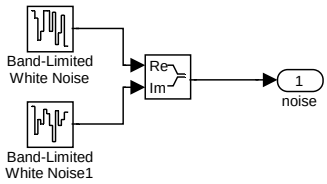


Figure 5.6: Complex bandlimited white Gaussian noise model (see [3])

The *complex bandlimited white noise* block contains two of Matlab's *bandlimited white noise* models (see [3] and figure 5.6). These two models are combined in such way that they form *complex noise*. The random noise values are added to the transmitted signal.

First, the *bit energy* E_b is determined for all modulation types. This value will be used later on to calculate the N_0 value that is necessary to reach a certain E_b/N_0-ratio. Although burst type in the models is set to *broadcast*, the

[4] See section 3.6.

[5] This equals $1.2\,10^6$ – $7.2\,10^6$ bits. If we want at least 100 bit errors per simulation, the results with a BER of $1\,10^{-5}$ or lower become unreliable.

energy in the preamble is not used to calculate E_b, because the calculations in section 2.10 assume that no preamble is used.

Next, BERs are measured with the *error rate calculation* block (see [7]), for certain E_b/N_0-ratios. In [7] section A.6 the Matlab script is printed, that is used to gather the results. The following sections will present the simulation results.

In the receiver model frequency offset estimation, phase offset estimation, channel equalization and symbol window drift tracking are disabled.

5.3.2 AWGN channel simulation results using 64-bit floating point numbers

In figure 5.7 the results of the AWGN channel experiment are compared to the theoretical performance of the system (explained in section 2.10). The receiver simulation model used 64-bit floating point numbers. This build-in *C++* number type uses ten exponent bits and 52 (plus one hidden) mantissa bits and two sign bits (see [4] and section 3.6.4).

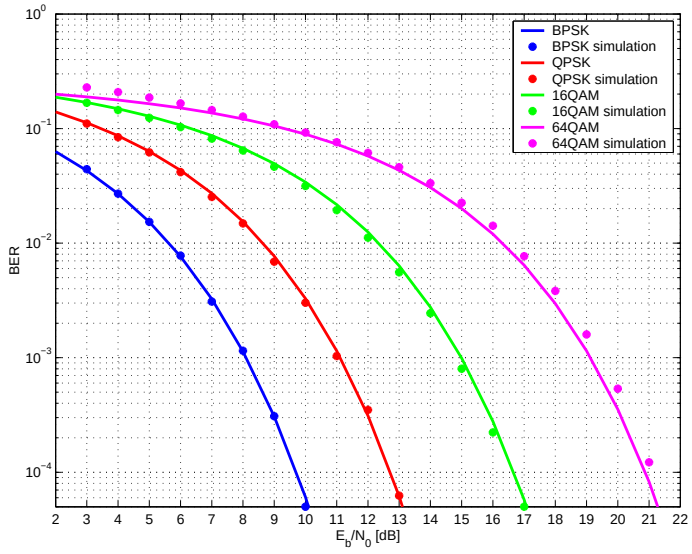


Figure 5.7: AWGN channel simulation BER results (64-bit floating point numbers)

The BPSK and QPSK simulation results match their theoretical expected BER E_b/N_0 relation. The 16QAM subcarrier modulation technique in the model performs slightly better than expected and 64QAM performs slightly worse than expected. Both stay within 0.1 dB to their expected "*waterfall*" curves.

The simulation results are used to calculate at what E_b/N_0-ratio the system meets its minimum sensitivity demand after FEC decoding (see sections 2.2.3 and 2.4). The results of this calculation are shown in table 5.3. The results show, that the 64QAM result differs 1 dB with the expected value. The other results

stay within 0.5 dB of their expected value.

Table 5.3: Simulated minimum E_b/N_0 requirements to reach a PER of 10% using packet length of 54 bytes. table 2.15 shows the *expected* values.

Bit-rate mode	Sub carrier modulation	R_c	Minimum E_b/N_0 [dB]
A	BPSK	1/2	3.5
B	BPSK	3/4	5.5
C	QPSK	1/2	6.0
D	QPSK	3/4	8.5
E	16QAM	9/16	11.0
F	16QAM	3/4	12.0
G	64QAM	3/4	16.0

5.3.3 AWGN channel simulation results using 32-bit fixed point numbers

The AWGN channel experiment is repeated with a receiver model, that uses 32-bit fixed point numbers. Fixed point numbers are discussed in section 3.6.3. One of the 32-bits is the *sign* bit. The point is said to be at position $g = 15$ (see section 3.6.1 for the definition of *position of the point*). In figure 5.8 the BER versus E_b/N_0 simulation results are plotted.

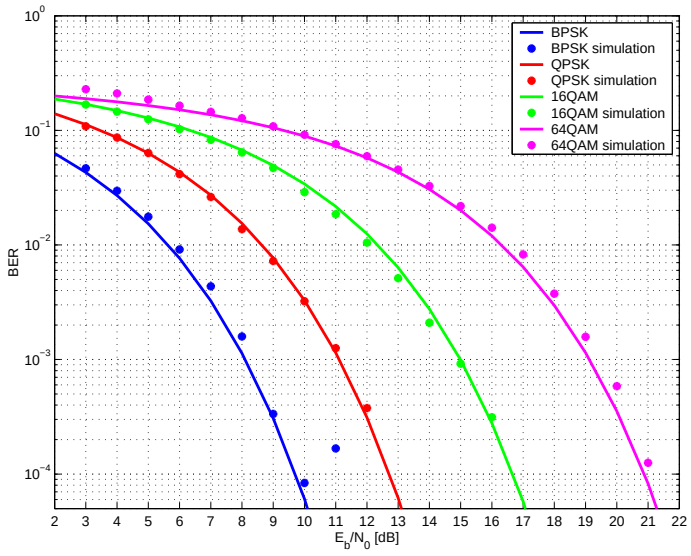


Figure 5.8: AWGN channel simulation results (32-bit fixed point numbers)

These simulation results are not significantly different with those found in section 5.3.2. Hence I agree with the choice made in [17] to use 32-bit fixed point numbers in the receiver implementation. Future work should investigate the effects of using 32-bit fixed point numbers in the digital channel selection filters.

5.3.4 AWGN channel simulation results using 16-bit fixed point numbers

When the AWGN channel experiment is repeated with 16-bit fixed point numbers, the results start to differ from their expected values. As can be expected, 64QAM suffers most from the computational noise, that is added in the model by using 16-bit fixed point numbers. Apparently the computational noise is the main disturbing factor from $E_b/N_0 \approx 16$ dB. The simulation results are plotted in figure 5.9.

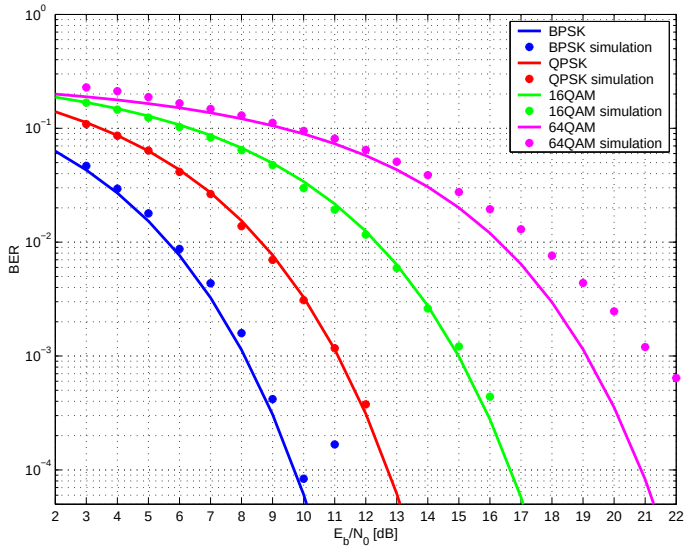


Figure 5.9: AWGN channel simulation results (16-bit fixed point numbers)

5.4 Phase offset simulation

In this section the functioning of the phase offset correction will be show on an AWGN channel. This function will also be used in the next experiment: the frequency offset experiment.

5.4.1 Experiment configuration

For this experiment the model presented in figure 5.5, will be extended with a *phase rotation* of the transmitted signal before noise is added and the signal is fed into the receiver. The model layout is depicted in figure 5.10.

The subcarrier modulation type is set to 16QAM and the burst type to broadcast. Per simulation 25,000 OFDM symbols will be transmitted. The all corrections functions in the receiver are disabled. In the model the *phase* is changed from 0 rad to π rad in steps of $\pi/4$ rad. For each phase offset the

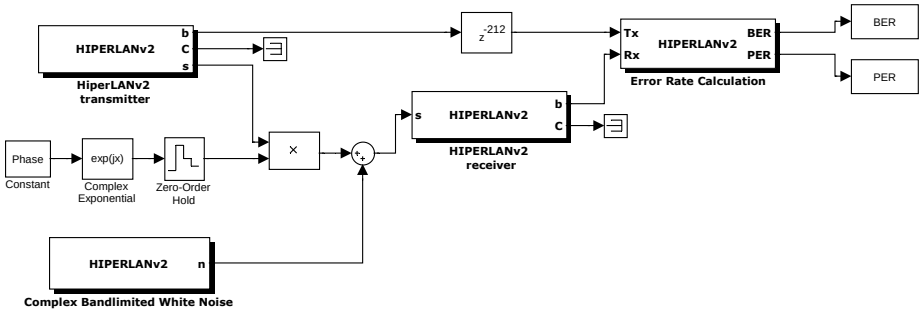


Figure 5.10: Model used for phase offset correction experiment

AWGN experiment is done. Finally the experiment is repeated with phase offset correction enabled.

5.4.2 Results

The results of the above described phase offset experiment are shown in figures 5.11 and 5.12. From the figures can be concluded that a phase offset causes a large BER, when it is not corrected. The phase offset corrector corrects without a problem the phase offsets for all E_b/N_0-ratios to at most 1 dB distance to the theoretical expected curve.

5.5 Frequency offset simulation results

A frequency difference between the mix frequency in transmitter and receiver cause inter-subcarrier interference (see section 3.3.1). Section 4.6 presented a method to make an *estimation* of the frequency offset. In this section the influence of noise on the frequency offset estimation and correction will be discussed by looking at simulation results.

5.5.1 Experiment configuration

To determine the influence of noise on the frequency offset estimation and correction, a frequency offset will be introduced to the input of the receiver by multiplying the output of the transmitter with a complex power of e. This is described in section 3.3.1, equation 3.23. The model configuration used for the experiment in this section is shown in figure 5.13.

The subcarrier modulation type is set to 16QAM and the burst type to broadcast. Per simulation $25,000$ OFDM symbols will be transmitted. The frequency and phase offset corrections are disabled. The frequency offset in the model is changed from $f_\Delta = 0$ to $f_\Delta = \Delta_f$ in steps of $\Delta_f/5$. For each frequency

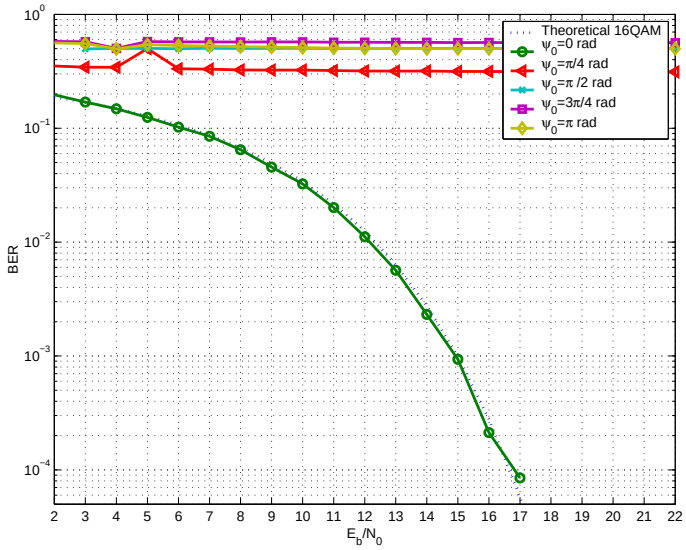


Figure 5.11: Phase offset simulation results. The phase offset correction is *not* active in this figure. Only $\psi_0 = 0$ rad is demodulated correctly. The other phase offsets case a BER of 0.5

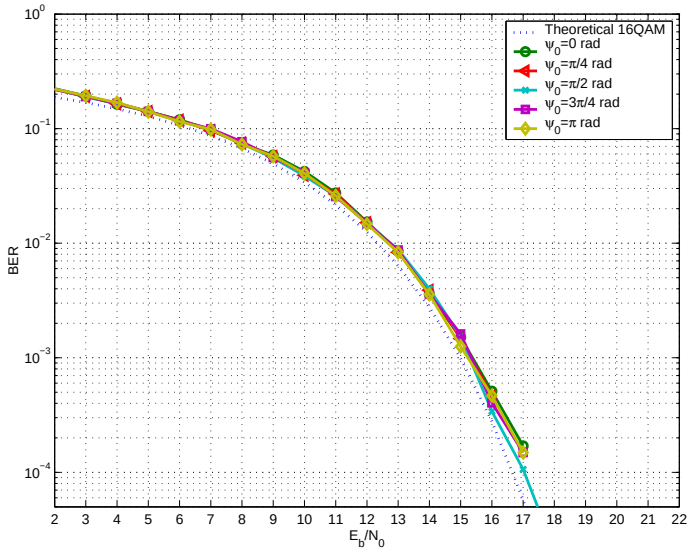


Figure 5.12: Phase offset simulation results. In this figure the phase offset correction is active. All phase offset results stay within 1 dB to the theoretical performance

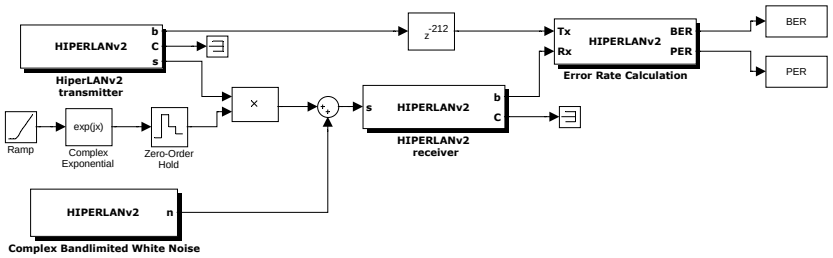


Figure 5.13: Model for frequency offset simulations. The "*ramp*" block is used to generate the time dependent argument for the complex power of e.

offset the AWGN experiment is done.[6] Finally the experiment is repeated with frequency and phase offset corrections enabled.

5.5.2 Results

In figures 5.14 and 5.15 the results of the above discussed experiment are depicted. From the figure 5.14 can be concluded that frequency offsets up to $f_\Delta = 62.5$ KHz do not need to be corrected; they do not have an influence on the performance of the system. For that frequency offset[7] the inter-subcarrier interference will cause that about 5% of the power of a neighboring subcarrier will spill into a subcarrier (see figure 3.10). Other frequency offsets cause a BER of 0.5.

Figure 5.15 shows the *waterfall* curves when the frequency offset correction and phase offset correction are enabled. The frequency offset corrector can correct all tested frequency offsets, except for $f_\Delta = \Delta_f$. Note that the maximum allowed frequency offset in the HiperLAN/2 system is 250 KHz (see section 3.3.1). The frequency offset results stay within 1 dB of their theoretical expected value.

In fact we test the influence of noise and frequency offset on the detection of the *preambles* with this experiment too. In figure 5.15 we see for low values of E_b/N_0 that sometimes the preamble detection fails. This results immediately in a BER of 0.5. Apparently, a frequency offset in combination with high noise levels, can cause that preamble section A is not recognized properly.

5.6 Signal generator – scope channel results

In this "*real world*" experiment a signal generator (Agilent E4438C vector signal generator) is used to transmit the HiperLAN/2 transmitter's complex time samples at baseband level. A scope (Tektronix TDS7407) was used to sample the transmitted baseband signal. The resulting samples were fed to the HiperLAN/2 receiver model and demodulated.

[6] Only for 64-bit floating point numbers.
[7] $f_\Delta/\Delta_f = 0.2$

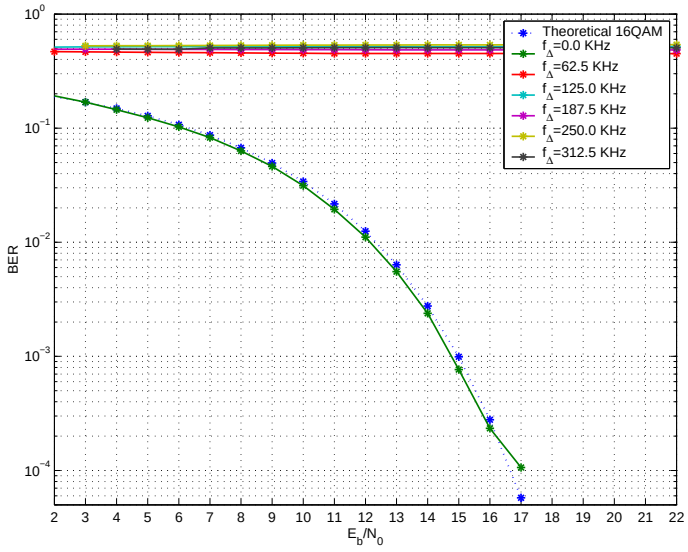


Figure 5.14: Frequency offset simulation results. The frequency offset correction is *not* active in this figure. $f_\Delta = 62.5$ KHz is demodulated correctly. Other frequency offsets cause BERs of ≈ 0.5

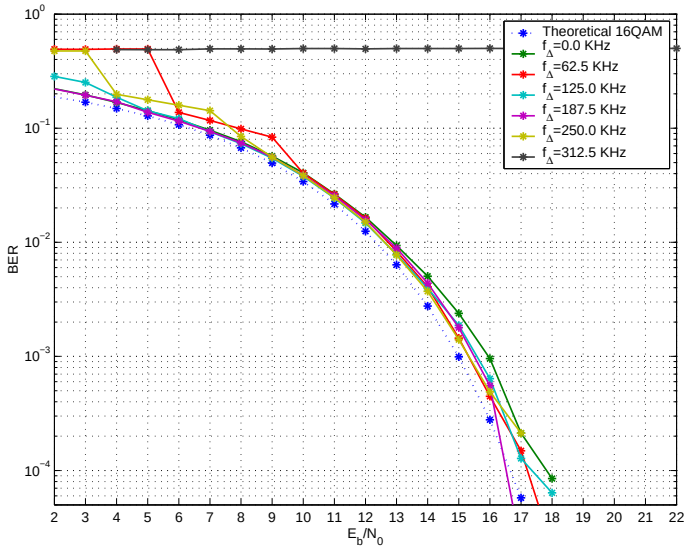


Figure 5.15: Frequency offset simulation results. In this figure the frequency correction is active. Most frequency offset results stay within 1 dB to the theoretical performance. $f_\Delta = \Delta_f = 312.5$ KHz can not be corrected.

The experiment configuration of this test is described in [28]. The signals that are generated by the Agilent signal generator, are sampled by the Tektronix scope at 80 MHz and a resolution of 8 bits.

The sampled values are imported in Matlab Simulink, down sampled to 20 MHz and fed to the receiver model. The experiment is repeated for BPSK, QPSK, 16QAM and 64QAM.

5.6.1 Results

In table 5.4 the measurement results are show of the experiment discussed above and in [28]. The BPSK and QPSK bits are demodulated without problem. The 16QAM and 64QAM bits are retrieved with the bit errors. Turning on the channel estimation in the receiver model, lowers the number of bit errors clearly.

Table 5.4: Results of the signal generator-scope experiment. About 1 ms of signal (250 OFDM symbols) was transmitted

Modulation technique	BER	Status channel equalization
BPSK	0.000	Enabled/Disabled
QPSK	0.000	Enabled/Disabled
16QAM	0.169	Disabled
16QAM	0.010	Enabled
64QAM	0.239	Disabled
64QAM	0.079	Enabled

With figure 5.7 we can conclude that the E_b/N_0-ratio in this experiment was ≈ 12.5 dB.

5.7 Conclusion

In this chapter some model simulation results have been obtained:

- We looked at a simulation using an ideal channel. This gave insight in how signals should look without distortions.
- The computational requirements were determined. Future work should study whether the computational requirements can be met by a general purpose processor. This type of processor is proposed in the SDR project [2] for signal operations. The demapping should be optimized in future work, to reduce its computational requirements.
- Next the AWGN channel experiment was repeated for three number types: 64-bit floating point, 32-bit fixed point and 16-bit point numbers. The 64-bit and 32-bit results matched the theoretical performance very closely. Thus 32-bit fixed point numbers can be used in a demodulator, as was

proposed in [17]. The 16-bit results start to differ from the theoretical expected values. As could be expected, 64QAM is most sensitive to distortion.

- The phase offset experiment showed, that phase errors –when they are not corrected– cause large bit errors. The phase offset corrector proposed in this report has an good performance. In corrects all phase offsets on an AWGN channel within 1 dB of theoretical expected value.

- A frequency offset causes inter-subcarrier interference. This was clearly visible in the simulation results of the frequency offset experiment. Only a frequency offset of $f_\Delta/\Delta_f = 0.2$ did not need to be corrected. The frequency offset corrector, proposed in this report, corrected the waterfall curves to at most 1 dB of the theoretical expected curves. A frequency offset of $f_\Delta = \Delta_f$ could not be corrected. Note that the maximum allowed frequency offset is 250 KHz. Sometimes the recognition of preamble section A failed.

- In the signal generator-scope experiment BPSK and QPSK could be demodulated without bit errors. Bit errors occurred when the experiment was repeated for 16QAM and 64QAM. The resulting BER could be lowered by enabling the channel estimator. Apparently the E_b/N_0-ratio in this experiment was ≈ 12.5 dB.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

In this Master of Science thesis a model of the WLAN HiperLAN/2 standard physical layer is presented. The model is intended for use in the SDR project, that is currently investigating the feasibility of a software defined radio by designing a *demonstrator*. The model consists of two parts: a HiperLAN/2 transmitter and a HiperLAN/2 receiver.

First the HiperLAN/2 physical layer in the transmitter was investigated:

- Input bits of the physical layer are scrambled. This causes a possible improvement of the BER.
- A convolutional encoder in combination with bit-rate mode dependent puncturing is used to apply FEC coding to the scrambled bits. The performance of the FEC coding in combination with an ideal decoder was analytical determined. The result is shown in figure 2.4.
- The FEC coded bits are interleaved to transmit neighboring bits on different subcarriers. This improves the BER in fading channel environments.
- The bits are mapped using BPSK, QPSK, 16QAM or 64QAM on the 48 data subcarriers used in the system. The average power of the transmitted signal is kept equal for all modulation types.
- Four pilot carriers are inserted to assist coherent demodulation.
- OFDM is used as modulation technique. The subcarriers are orthogonal to each other in the *useful symbol part*. OFDM can efficiently be implemented with an IFFT function.

- A cyclic prefix is added to the signal, to combat inter-symbol interference caused by the radio channel.
- The theoretical performance of the HiperLAN/2 OFDM system on an AWGN channel was outlined in section 2.10. The results are depicted in figures 2.11 and 2.12 and table 2.15. These results can be used to test a decoder.
- A HiperLAN/2 physical layer simulation model of the transmitter has been implemented. It models the mapping, OFDM and physical burst generation functions. It passed its functional test.

The transmitted signal is distorted by the radio channel, the receiver hardware and the channel selection filters before it reaches the in software implemented demodulation part in the receiver.

- Reflection, scattering and diffraction cause the receiver to receive attenuated and delayed versions of the transmitted signal. This is modelled with the Rayleigh channel model.
- The maximum delay spread in the HiperLAN/2 transmission band is typically 130 ns. The cyclic prefix of an OFDM symbol is long enough to combat inter-symbol interference caused by the delay spread.
- A *channel equalizer* should be implemented in a HiperLAN/2 demodulator. It should be adaptive and it should be updated at least once in 10 ms. The channel can be described at baseband level, thus the channel equalizer may also operate at baseband level.
- Impurities in the mixer frequencies, that are used in the analog parts of a HiperLAN/2 transmitter and HiperLAN/2 receiver can be divided in: frequency offset, common phase offset and phase noise. A frequency offset causes local inter-subcarrier interference locally, while phase noise causes inter-subcarrier interference with all subcarriers. Common phase offset causes a phase rotation equal to all subcarriers. These effects can cause large BERs. The receiver should estimate the distortions and compensate for them. Note that the maximum allowed frequency offset is 250 KHz (see [8]).
- The analog to digital converter introduces quantization noise. The anti-aliasing filter distorts the subcarrier values.
- A difference in sample length between HiperLAN/2 transmitter and receiver causes symbol window drift. This cause a phase rotation of the subcarrier values that is dependent on the subcarrier number.
- The digital channel selection filter gives rise to errors in the subcarrier values. Since these disturbances are known, the receiver can easily correct the errors.
- The receiver uses an ALU to perform calculations. The ALU has a finite precision. This causes computational noise to signals in the receiver. A

model of an ALU has been presented, that can simulate computational noise.

The following demodulation functions are implemented in the receiver model:

- *Serial to parallel conversion.* This function is implemented in the model, because the main demodulation function of the receiver, an IDFT, works with parallel data instead of the serial data generated by the hardware of the receiver.
- *Synchronization.* This function takes care of detecting a transmission burst, detecting the preambles in the burst and tracking symbol window drift. Demodulation cycles will take place about every 80 received samples.
- *Prefix removal.* After synchronization the useful data part is extracted from the input samples.
- *Frequency offset correction.* An estimation of frequency offset is made using the preambles in the burst. This estimation is used to rotate all input samples of the receiver appropriately.
- *Inverse OFDM.* The subcarrier values are retrieved from the time domain samples by applying the efficient FFT algorithm.
- *Channel equalization.* The implemented receiver model uses preamble section C to determine the channels transfer function. This transfer function is obtained and equalized in the frequency domain.
- *Demapping.* A received subcarrier value is quantized into one of all possible subcarrier values of a mapping technique, by determining the smallest Euclidean distance. Next the bit values associated with the symbol are passed to the output of the receiver model. This is repeated for all 48 subcarriers.

The HiperLAN/2 models are implemented in Matlab Simulink (see [3]) using C++ as programming language. The models are tested in [28] and [16]. They passed their functional test. The following experiments are conduced in this report:

- The computational requirements were determined. The results are presented in tables 5.1 and 5.2.
- Three AWGN channel experiments were conducted for three number types: 64-bit floating point, 32-bit fixed point and 16-bit point numbers. The 64-bit and 32-bit results matched the theoretical performance very closely (0.1 dB). Thus 32-bit fixed point numbers can be used in a demodulator, as was proposed in [17]. The 16-bit results start to differ from the theoretical expected values. As could be expected, 64QAM is most sensitive to distortion.

- The phase offset experiment showed, that phase errors –when they are not corrected– cause large bit errors. The phase offset corrector proposed in this report has an good performance. In corrects all phase offsets on an AWGN channel within 1 dB of theoretical expected value.
- The frequency offset experiment showed, that frequency offsets –when they are not corrected– cause large bit errors. Only a frequency offsets up to $f_\Delta/\Delta_f = 0.2$ do not need to be corrected. The frequency offset corrector, proposed in this report, corrected the waterfall curves to within 1 dB of the theoretical expected curves. A frequency offset of $f_\Delta = \Delta_f$ could not be corrected.
- In the signal generator-scope experiment BPSK and QPSK could be demodulated without bit errors. Bit errors occurred when the experiment was repeated for 16QAM and 64QAM. The resulting BER could be lowered by enabling the channel estimator. Apparently the E_b/N_0-ratio in this experiment was ≈ 12.5 dB. With this experiment is also proved that the model can generate test signals and that the model can demodulate captured signals.

6.2 Recommendations

The following recommendations are made in this report:

- Scrambling, FEC coding and interleaving functions should be implemented in the HiperLAN/2 physical layer models.
- The five channel models created by ETSI for use in HiperLAN/2 simulations (see [21]) should be implemented.
- Since the coherence time of the channel is about 10 ms (see section 3.2.4) and a burst has a duration of 2 ms, an adaptive channel equalizer was not implemented. Simulations with the channel models must show that this is a correct approach.
- The demapping function is responsible for a large part of the computational requirements of the receiver. A future study must investigate whether the demapping function can be implemented more efficiently.
- The demonstrator of the SDR project will be using a *general purpose processor* of a personal computer to perform the demodulation tasks in the digital domain (see [2]). Future studies must show whether the proposed processor can meet the computational requirements of the receiver algorithms discussed in this report.
- Future work should investigate the effects of using 32-bit fixed point numbers in the digital channel selection filters on the performance of the system.

Bibliography

[1] F.W. Hoeksema R. Schiphorst C.H. Slump. *Functional Analysis of a SDR Based Bluetooth/HiperLAN Terminal Demonstrator.*

[2] V.J. Arkesteijn R. Schiphorst. *Physical Layer Functions of a Software Radio based Bluetooth - HiperLAN/2 Demonstrator.* SDR project R002v01, July 2001.

[3] The Matworks Inc. *Matlab product documentation*, 2000. Version 6 Release 12.

[4] G. Laan. *Aan de slag met C++.* Uitgeverij Pim Oets, 2 edition, 1997.

[5] J. Walrand. *Communication Networks.* McGraw-Hill, U.S.A., second edition edition, 1998.

[6] M. Johnsson. *HiperLAN/2 – The Broadband Radio Transmission Technology Operating in the 5 GHz Frequency Band.* HiperLAN/2 Global Forum, 1999. http://www.hiperlan2.com.

[7] L.F.W. van Hoesel. *Design and implementation of a software defined HiperLAN/2 physical layer model for simulation purposes; Source code of the simulation model.* Master of Science Thesis, Universiteit Twente, EL SAS 032 N 02 a, 2002.

[8] ETSI. *Broadband Radio Access Networks (BRAN); HIPERLAN Type 2; Physical (PHY) layer.* 2001. ETSI TS 101 475 V1.2.2 (2001-02).

[9] ETSI. *Digital Video Broadcasting (DVB); Measurement Guidelines for BVB Systems.* 2001. ETSI TS 101 290 V1.2.1 (2001-05).

[10] S. Haykin. *An introduction to analog and digital communications.* John Wiley & Sons Inc. U.S.A., 1989.

[11] S. Hüttinger. *Investigation of hybrid ARQ for HiperLAN/2* . Lehrstuhl für Nachrichtentechnik, Friedrich-Alexander Universität Erlangen-Nürnberg, 1999.

[12] R.L. Peterson R.E. Ziemer. *Introduction to digital communication.* Prentice Hall, U.S.A., second edition, 2001.

[13] J.G. Proakis. *Advanced digital signal processing.* Macmillan, U.S.A., 1989.

[14] L.W. Couch. *Digital and analog communication systems.* Prentice Hall, U.S.A., fifth edition, 1997.

[15] J.G. Proakis. *Digital Communication.* McGraw Hill, 1989.

[16] L.F.W. van Hoesel. *Verification of the HiperLAN/2 transmitter.* SDR project R004v01, March 2002.

[17] V.J. Arkesteijn F.W. Hoeksema R. Schiphorst. *Specification and Design of a Software Radio based Bluetooth - HiperLAN/2 Demonstrator.* SDR project R004v01, April 2002.

[18] R. van Damme D. Dijkstra G. Still C. Traas G. Zwier. *Numerieke wiskunde.* Faculteit der toegepaste wiskunde, Universiteit Twente, The Netherlands, 1997.

[19] A.J.W.M. ten Berg J. Hofstede E. Molenkamp G.J.M. Smit. *College handleiding: Computer Organisatie.* Faculteit der informatica, Universiteit Twente, The Netherlands, 1996.

[20] S. Armour M. Butler A. Doufexi et al. *A Comparison of the HiperLAN/2 and IEEE 802.11a Wireless LAN Standards.* IEEE Communications Magazine, pages 172–180, May 2002.

[21] J. Medbo P. Schramm. *ETSI EP BRAN3ERI085B; Channel Models for HiperLAN/2 in Different Indoor Scenarios.* Not publical available yet, March 1998.

[22] R. Landman. *Wireless Indoor Channel Modeling for IEEE 802.11(b) System Simulation Purposes.* Master of Science Thesis, Universiteit Twente, EL-S&S-01.012, 2001.

[23] R. van Nee. *OFDM for wireless multimedia communications.* Artech House Publishers England, 2000.

[24] L. Hanzo T. Keller W. Web. *Single- and Multi-carrier Quadrature Amplitude Modulation; Principles and Applications for Personal Communications, WLANs and Broadcasting.* Wiley, U.S.A., 2000.

[25] A. Burr. *Modulation and coding for wireless communications.* Pearson Education, England, 2001.

[26] S. Wilson. *Digital modulation and coding.* Prentice Hall, U.S.A., 1996.

[27] A. Levesque K. Pahlavan. *Analysis of 60 GHz band indaar wireless channels with channel configuration.* IEEE International symposium on personal indoor and mobile radio propagation, 1998.

[28] V.J. Arkesteijn F.W. Hoeksema L.F.W. van Hoesel L.C. van Mourik R. Schiphorst. *Realization and Testing of a Software Radio based Bluetooth - HiperLAN/2 Demonstrator.* SDR project R005v01, July 2002.

[29] M. Breiling J.B. Huber S.H. Müller-Weinfurtner. *SLM Peak-Power Reduction Whithout Explicit Side Information.* IEEE Communications Letters, VOL. 5(NO. 6):239–241, June 2001.

[30] L.F.W. van Hoesel. *Indoor Radio Transmission using Orthogonal Frequency Division Multiplexing.* Universiteit Twente, S&S 020.01, November 2001.

[31] P. Kinget. *Integrated GHz Voltage Controlled Oscillators.* Bell Labs – Lucent Technologies.

[32] S.A. Fechtel G. Fock H. Meyr M. Speth. *Optimum receiver design for wireless broad-band systems using OFDM - Part I.* IEEE Transactions on Communications, Vol 47 No. 11 p. 1668-1677, 1999.

[33] F.W. Hoeksema. *Three Filters for HiperLAN/2 Channel Selection.* SDR project R006v01, July 2002.

[34] T. Heunks. *Atlas van de Wiskunde*, volume 2. Bosch & Keubing, The Netherlands, 1980.

[35] *Numerical Recipes in C: The Art of Scientific Computing; Fast Fourier Transform*, 2002. `http://www.ulib.org/webRoot/Books/Numerical_Recipes/bookcpdf/c12-2.pdf`.